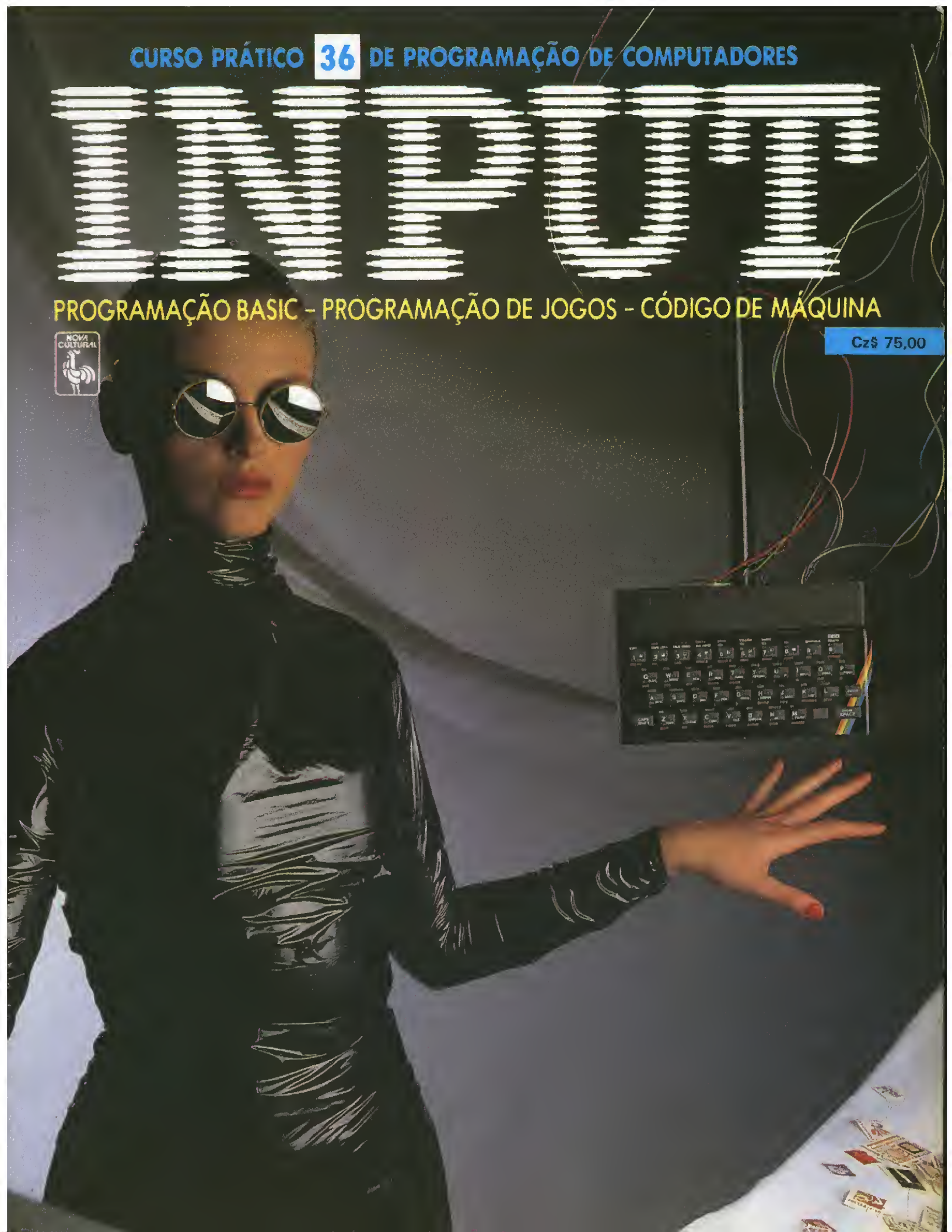
CURSO PRÁTICO 36 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

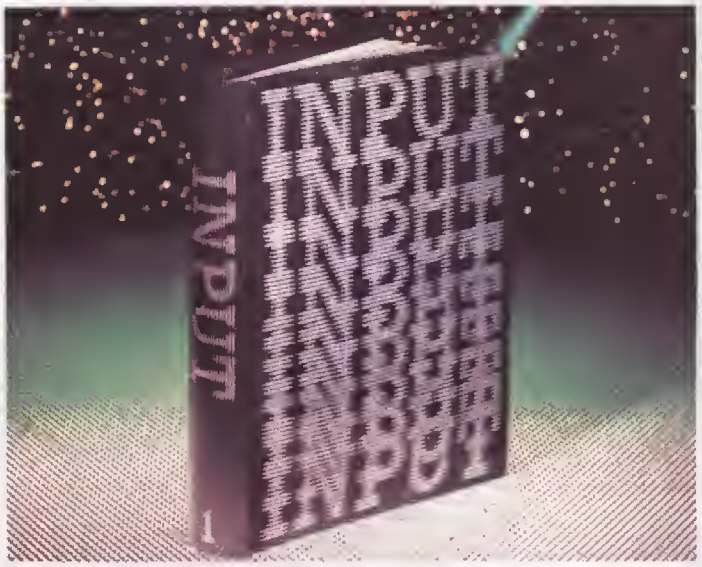
Cz$ 75,00



INPUT

Vol. 3 .N.º 36

NESTE NÚMERO



PLANO DA OBRA
INPUT é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

FÉRIAS, VIAGENS, MUDANÇAS...
NÃO FIQUE COM A COLEÇÃO INCOMPLETA
Se você está saindo de férias, pretende viajar ou vai se ausentar por algum tempo, avise antecipadamente seu jornaleiro. Ele pode guardar os seus fascículos enquanto você estiver fora. Se, por qualquer motivo, você perdeu alguns números, peça-os também a seu jornaleiro, ou entre em contato com nossa Distribuidora:

1. **Pessoalmente** — Em *São Paulo*, os endereços são: Rua Brigadeiro Tobias, 773, Centro, telefone 227-4188; Av. Industrial, 117, Santo André, telefone 449-0411, das 7h30 às 17h00 - dias úteis. No *Rio de Janeiro*, Av. Mem de Sá, 191/193, Centro, telefone (021) 222-7422, das 7h30 às 17h00 - dias úteis.
2. **Por carta** — Envie para:
   DINAP — Distribuidora Nacional de Publicações
   Números Atrasados
   Estrada Velha de Osasco, 132 — Jardim Teresa
   CEP 06040 — Osasco — SP
3. **Por telex** — Utilize o nº (11) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Lda. — Qta. Pau Várais, Azinhaga de Fetais, 2685, Camarate, Lisboa; Apartado 57; Telex 43 069 JARLIS P.
Atenção: Após seis meses do encerramento da coleção, o atendimento dos pedidos dependerá da disponibilidade do estoque.
Obs.: Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

COLABORE CONOSCO
Encaminhe seus comentários, críticas, sugestões ou reclamações ao
SERVIÇO DE ATENDIMENTO AO LEITOR
Caixa Postal 9 442, São Paulo — SP.

EDITOR
RICHARD CIVITA
NOVA CULTURAL

Presidente
Flávio Barros Pinto
Diretoria
Carmo Chagas, Iara Rodrigues
Pierluigi Bracco, Plácido Nicoletto,
Roberto Silveira, Shozi Ikeda,
Sônia Carvalho

REDAÇÃO
Diretor Editorial: Carmo Chagas
Editores Executivos:
Stefania Crema, Berta Sztark Amar
Editor Chefe: Paulo de Almeida
Editoras Assistentes: Ana Lúcia B. de Lucena,
Marisa Soares de Andrade
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio,
Grace Alonso Arruda, Monica Lenardon Corradi

Secretário de Redação: Mauro de Queiroz

Colaboradores
Consultor Editorial Responsável:
Dr. Renato M. E. Sabbatini
(Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas - SP
Tradução, adaptação, programação e redação:
Abílio Pedro Neto, Aluísio J. Dornellas de Barros,
Marcelo R. Pires Therezo, Marcos Huascar Velasco,
Raul Neder Porrelli, Ricardo J. B. de Aquino Pereira.
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor Comercial: Roberto Silveira
Gerente Comercial: Joaquim Celestino da Silva
Gerente de Circulação: Denise Mozol
Gerente de Propaganda e Publicidade: José Carlos Madio
Gerente de Pesquisa e Análise de Mercado:
Wagner M. P. Nabuco de Araújo

CLC

A Editora Nova Cultural Ltda. é uma empresa do Grupo CLC — Comunicações, Lazer, Cultura S.A.
Presidente: Richard Civita
Diretoria: Flávio Barros Pinto, João Gomez,
Menahem M. Politi, Renê C.X. Santos,
Stélio Alves Campos

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta pela AM Produções Gráficas Ltda. e impressa pela Companhia Lithographica Ypiranga.

ADIVINHAÇÃO DE PALAVRAS

- UM DESAFIO PARA DOIS
- MONTAGEM DA TELA
- AS REGRAS DO JOGO
- OS VALORES DAS LETRAS
- ESTRATÉGIA

Este jogo de adivinhação de palavras agrada a todas as idades, pode ter o nível de dificuldade que se queira e é muito versátil. Inspirado no "jogo da forca", destina-se a dois jogadores.

Alguns jogos para computador não são apenas recreativos — como os de aventura —, prestando-se muito bem para fins educativos. É o caso do que apresentamos aqui, inspirado no conhecido "jogo da forca".

Este, adaptado ou não para o computador, ajuda não só a ampliar o vocabulário do jogador, assim como seus conhecimentos gerais sobre vários temas (já que é usual a definição prévia de um assunto sobre o qual as frases ou palavras versarão).

O jogo de INPUT, destinado a duas pessoas, também tem como objetivo a adivinhação de palavras ou frases. É mais interessante e mais divertido que o tradicional "jogo da forca" e tão educativo quanto ele. Pode ser jogado da mesma maneira, estabelecendo-se um assunto, o número de letras das palavras ou outro critério que se desejar.

O JOGO

Inicialmente, digite o nome dos dois jogadores. Em seguida, escolha o número de palavras da frase que cada jogador escreverá. Observe que, muitas vezes, frases mais longas são mais fáceis de se adivinhar, devido ao maior número de letras que aparecem.

Depois de ter escolhido o número de palavras, defina o número de jogadas que irão constituir a partida — isto é, quantas palavras cada jogador terá para adivinhar.

O primeiro jogador deve, então, pensar em uma frase e escrevê-la no computador. O adversário não precisa estar ausente enquanto você digita sua frase, pois as letras não aparecerão na tela. Mas, confiando em seu oponente, você poderá optar por ver as letras, o que lhe dará a certeza de que não cometeu nenhum erro de digitação.

Deixe apenas um espaço entre cada uma das palavras da frase. No caso de um único vocábulo, nenhum espaço é permitido. O tamanho máximo de cada frase é de quarenta caracteres no micro Apple e no TK-2000, 64 no Spectrum e no TRS-Color e 78 no MSX.

Quando terminar a digitação, pressione a tecla <**ENTER**> para que a tela principal apareça. Em seu topo, você dará, então, o nome dos jogadores e o número de pontos de cada um — 200, para começar o jogo.

Abaixo do placar, há uma tabela indicando os valores das letras. As de uso freqüente têm valor mais alto. As menos usuais têm um valor baixo. A frase a ser adivinhada aparece como uma seqüência de asteriscos.

No rodapé da tela encontram-se as instruções e, também, um espaço para os comandos e palpites.

ESTRATÉGIA

O jogador tem três opções de jogada: comprar letras (ou espaço), adivinhar uma letra em determinada posição, ou adivinhar a frase toda.

No início da jogada, uma boa opção é comprar um espaço — se suspeitar que a frase tem mais de uma palavra, é claro. Vogais são caras, mas aparecem com freqüência. As letras mais baratas aparecem muito menos e você corre o risco de não esclarecer nada com elas. Sempre é mais fácil chegar à frase correta quando se tem algumas consoantes — portanto, não se preocupe demais com as vogais.

Quando a frase começar a tomar forma, provavelmente você quererá adivinhar uma letra em determinada posição. Tendo uma palavra como S*U, por exemplo, a letra O será um bom palpite. É nesse momento que se pode ganhar pontos. Colocando uma letra na posição correta, você ganha o seu valor em pontos. Se errar, perde apenas metade do valor. Pressione XX para selecionar esta opção e dar o seu palpite.

Você pode, também, ter uma súbita inspiração e tentar acertar a frase toda. Nesse caso, deve pressionar ZZ e escrever a frase. Se ela estiver correta, o valor de todas as letras que ainda não tinham sido descobertas será acrescentado ao seu placar. Se você errar, perderá 50 pontos.

Digite a primeira parte do programa. Estas linhas fazem o computador acei-

tar todos os dados para iniciar o jogo. Mas com ela você não irá muito longe. A parte final — que contém as rotinas que lêem os palpites, fazem as correções e contam os pontos — fica para o próximo artigo.

Não se esqueça de gravar o programa.

```
10 LET R$="PALAVRA": LET W=14
: LET d=0: LET f=1: LET g$=""
: LET g=0: LET k=0: LET q$=""
: LET ta=200: LET tb=200: LET
tc=0: LET b=0: POKE 23609,50:
POKE 23658,8: LET i$="": LET
j$="": LET z$="": LET c$=""
20 FOR n=0 TO 7: READ y: POKE
USR "a"+n,y: NEXT n
30 DATA 255,129,129,129,129,
129,129,255
40 INPUT "NOME DO PRIMEIRO JO
GADOR ?       (ATE 7 LETRAS)",
 LINE a$
50 INPUT "NOME DO SEGUNDO JOG
ADOR ?        (ATE 7 LETRAS)"
, LINE b$
60 IF LEN a$>7 OR LEN b$>7
THEN  GOTO 40
70 CLS : INPUT "QUANTAS PALAV
RAS POR FRASE?(1-9)", LINE c$
 80 IF LEN c$<>1 THEN  GOTO 70
 90 IF CODE c$<49 OR CODE c$>
 57 THEN  GOTO 70
100 LET c=VAL c$
110 INPUT "NUMERO DE JOGADAS ?
(1 a 9)", LINE t$
120 IF LEN t$<>1 THEN  GOTO
110
130 IF CODE t$<49 OR CODE t$>
57 THEN  GOTO 110
140 LET t=VAL t$
150 IF c>1 THEN  LET j$="S":
LET i$="COM UM ESPACO ENTRE CA
DA": LET r$="FRASE"
160 PRINT a$;",E SUA VEZ DE JO
GAR."''"INTRODUZA SUA FRASE DE
";c;" PALA - VRA";j$;".AS LETR
AS QUE VOCE INTRODU-ZIR SERAO
INVISIVEIS,MAS SE VOCEQUISER V
ER ENTAO PRESSIONE '0'. PARA C
ONTINUAR, PRESSIONE 1."
170 LET k$=INKEY$: IF k$=""
THEN  GOTO 170
190 IF k$="0" THEN  POKE 23624
,56: INPUT  LINE a$: CLS :
GOTO 220
200 IF k$="1" THEN  POKE 23624
,63: INPUT  LINE a$: CLS :
POKE 23624,56: GOTO 220
210 GOTO 170
220 LET l=LEN a$
230 IF l=0 THEN  PRINT "ENTRAD
A ILEGAL. TENTE DE NOVO":
PAUSE 100: CLS : GOTO 160
240 IF l>64 THEN  PRINT "ENTRA
DA MUITO LONGA. TENTE DE  NOV
O": PAUSE 100: CLS : GOTO 160
250 FOR n=1 TO l: IF a$(n)=
CHR$ 32 THEN  LET d=d+1: GOTO
270
260 IF CODE a$(n)<65 OR CODE
a$(n)>90 THEN  PRINT "LETRA IL
EGAL. TENTE DE NOVO": PAUSE
100: CLS : LET d=0: GOTO 160
270 IF c=1 AND d=1 THEN  PRINT
"NAO HA ESPACOS DENTRO DE UMA
   UNICA PALAVRA. TENTE DE NOV
O.": PAUSE 100: CLS : LET d=0:
 GOTO 160
280 NEXT n
290 IF d<>c-1 THEN  PRINT "VOC
E DEVE INTRODUZIR ";c;" PALAVR
AS ";i$;". TENTE OUTRA VEZ":
PAUSE 100: CLS : LET d=0: GOTO
 160
300 LET z$=""
310 FOR n=1 TO l: LET z$=z$+"*
": NEXT n
320 PRINT  INK 1;AT 0,0;"SCORE
/";a$: PRINT  INK 1;AT 0,16;"S
CORE/";b$: PRINT  PAPER 2, INK
6;AT 1,6;ta;TAB 22;tb;TAB 31;"
 "
330 PRINT AT 3,7;"VALOR DOS CA
RACTERES"
340 FOR n=0 TO 26: READ q$:
LET q$=q$+q$: NEXT n: PRINT q$
: RESTORE 900
350 PRINT  INK 1;AT 12,0;"A ";
r$;" QUE ";b$;" TEM QUE  ADIVI
NHAR CONTEM ";L;" CARACTERES":
PRINT  PAPER 2; INK 6;z$
360 INPUT "VOCE QUER COMPRAR U
M CARACTER   PELO PRECO EXIBID
O NA TABELA?   FACA A ESCOLHA
DO CARACTER.     SENAO, TECLE
XX PARA ADIVINHAR  UM CARACTER
 OU ZZ PARA ADIVINHARA FRASE T
ODA.", LINE d$
1000 DATA "A-20    ","B-10    "
,"C-10    ","D-12    ","E-20
 ","F-08    ","G-12    ","H-08
  "
1010 DATA "I-20    ","J-04    "
,"K-06    ","L-10    ","M-10
 ","N-10    ","O-20    ","P-10
  ","Q-02    ","R-12    ","S-1
2    "
1020 DATA "T-12    ","U-20    "
,"V-08    ","W-08    ","X-04
 ","Y-08    ","Z-02    ","<GRAP
HICS A>-20    "
```

```
5 CLEAR 1000
10 R$="PALAVRA":W=14:F=1:TA=200
:TB=200
15 P1=PEEK(359):P2=PEEK(360):P3
=PEEK(361)
40 CLS:LINE INPUT"NOME DO PRIME
IRO JOGADOR          (MAX 7 LETRA
S) ?";A$
50 PRINT:LINE INPUT"NOME DO SEG
UNDO JOGADOR          (MAX 7 LET
RAS) ?";B$
60 IF LEN(A$)>7 OR LEN (B$)>7 T
```

MICRO DICAS

MANIPULAÇÃO DE CORDÕES

Os comandos BASIC capazes de agir sobre variáveis alfanuméricas constituem a chave do sucesso do programador interessado em jogos e aplicativos que manipulem palavras.

Comandos como **RIGHT$**, **LEFT$** e **MID$** permitem a "leitura" do conteúdo de uma cadeia de caracteres. E temos ainda funções como **LEN**, **CHR$**, **STR$** e **STRING$**, que tornam possível a criação de novos cordões ou a transformação dos antigos.

Observe que os comandos seguidos por um cifrão —"$"— geram novos cordões, enquanto os comandos sem este sinal resultam num valor numérico que corresponde ao operando.

```
HEN 40
70 CLS:LINE INPUT"ESCOLHA O NIV
EL DE DIFICULDADE  (NUMERO DE P
ALAVRAS/FRASE 1-9) ?";C$
80 IF LEN(C$)<>1 THEN 70
90 IF C$<"1" OR C$>"9" THEN 70
100 C=VAL(C$)
110 PRINT:LINE INPUT "NUMERO DE
 JOGADAS (1-9) ? ";T$
120 IF LEN(T$)<>1 THEN  110
130 IF T$<"1" OR T$>"9" THEN 11
0
140 T=VAL(T$)
150 IF C>1 THEN J$="S":I$="COM
UM ESPACO ENTRE CADA":R$="FRASE
"
155 CLS
160 PRINT A$;", E SUA VEZ DE JO
GAR":PRINT:PRINT"INTRODUZA SUA
FRASE DE ";C;" PALA-VRA";J$
165 PRINT:PRINT"PRESSIONE'0'PAR
A VER AS LETRAS  OU '1' PARA CO
NTINUAR ...":PRINT
170 K$=INKEY$:IF K$="" THEN 170
190 IF K$="1" THEN PRINT "? ";:
POKE 359,&H86:POKE 360,32:POKE
361,57:LINE INPUT S$:POKE 359,P
1:POKE 360,P2:POKE 361,P3:GOTO
220
200 IF K$="0" THEN LINE INPUT "
? ";S$:GOTO 220
210 GOTO 170
220 L=LEN(S$):PRINT
230 IF L=0 THEN PRINT "ENTRADA
ILEGAL - TENTE DE NOVO":GOSUB 9
50:CLS:GOTO 160
240 IF L>64 THEN PRINT"ENTRADA
MUITO LONGA-FACA DE NOVO":GOSUB
 950:CLS:GOTO 160
250 FOR N=1 TO L:IF MID$(S$,N,1
)=CHR$(32) THEN D=D+1:GOTO 270
260 IF MID$(S$,N,1)<"A" OR MID$
(S$,N,1)>"Z" THEN PRINT "CARACT
ER ILEGAL - TENTE DE NOVO":GOSU
B 950:CLS:D=0:GOTO 160
270 IF C=1 AND D=1 THEN PRINT"N
AO HA ESPACOS DENTRO DE UMA
UNICA PALAVRA! TENTE DE NOVO":G
OSUB 950:CLS:D=0:GOTO 160
280 NEXT N
290 IF D<>C-1 THEN PRINT "VOCE
PRETENDE INTRODUZIR";C;"PALA-VR
AS ";I$:PRINT"TENTE DE NOVO":GO
SUB 950:CLS:D=0:GOTO 160
300 Z$=""
310 FOR N=1 TO L:Z$=Z$+"*":NEXT
 N
320 CLS:PRINT"SCORE/";A$,"SCORE
/";B$:PRINT @38,TA;TAB(22);TB;"
  "
330 PRINT @69," VALOR DOS CARAC
TERES"
340 FOR N=0 TO 26:READ G$:Q$=Q$
+G$:NEXT N:PRINT Q$:RESTORE
350 PRINT @320,"A ";R$;" CONTEM
";L;"LETRAS":PRINT Z$
360 PRINT @416,"";:LINE INPUT "
XX=ADIVINHAR LETRA
 ZZ=ADIVINHAR A FRASE
  A-Z=COMPRAR O CARACTER   ?";D
$
900 DATA "A-20     ","B-10     ",
"C-10     ","D-12     ","E-20
","F-08     ","G-12     ","H-10
  "
910 DATA"I-20     ","J-04     ","
K-06     ","L-10     ","M-10     "
,"N-10     ","O-20     ","P-10
 ","Q-02     ","R-12     ","S-12
  "
920 DATA "T-12     ","U-20     ",
"V-08     ","W-08     ","X-04
",","Y-08     ","Z-02     ","e-20
 "
```

```
5 CLEAR 2000:KEYOFF:COLOR 15,4,
4
10 R$="palavra":W=14:F=1:TA=200
:TB=TA
40 CLS:LINEINPUT"Nome do jogado
r 1 (max 9 letras) ?      ";A$
50 PRINT:LINE INPUT"Nome do jog
ador 2 (max 9 letras) ?      ";B
$
60 IF LEN(A$)>9 OR LEN(B$)>9 TH
EN 40
70 CLS:PRINT"Qual o nível de di
ficuldade":INPUT"(número de pal
avras 1-9)";C$
90 IF C$<"1" OR C$>"9" THEN 70
100 C=VAL(C$)
110 PRINT:INPUT"Número de jogad
as (1-9)";T$
130 IF T$<"1" OR T$>"9" THEN 11
0
140 T=VAL(T$)
150 IF C>1 THEN J$="s":I$="com
um espaço entre elas":R$="FRASE
"
160 CLS:PRINTA$;", é a sua vez.
```

```
":PRINT"Digite sua frase de";C;
"palavra";J$;"."
165 PRINT:PRINT"Se desejar ver
as letras digitadas, pressione
<0>, senão pressione <1>.";
170 K$=INKEY$:IF K$="" THEN 170
175 IF K$<>"1" AND K$<>"0" THEN
 170
180 IF K$="0" THEN 200 ELSE PRI
NT:PRINT"? ";
190 K$=INKEY$:IF K$="" THEN 190
195 IF K$<>CHR$(13) THEN S$=S$+
K$:GOTO 190 ELSE 220
200 PRINT:LINEINPUT"? ";S$
220 L=LEN(S$):PRINT
230 IF L=0 THEN PRINT"Entrada i
legal - repita":GOSUB 950:CLS:G
OTO 160
240 IF L>78 THEN PRINT"Entrada
muito longa - repita":GOSUB 950
:CLS:GOTO 160
250 FOR N=1 TO L:IF MID$(S$,N,1
)=CHR$(32) THEN D=D+1:GOTO 270
260 IF MID$(S$,N,1)<"A" OR MID$
(S$,N,1)>"Z"ANDMID$(S$,N,1)<>"Ç
"THEN PRINT"Caracter ilegal - r
epita":GOSUB 950:CLS:D=0:GOTO 1
60
270 IF C=1 AND D=1 THEN PRINT"E
spaços não são permitidos em um
a      única palavra - repita":
GOSUB 950:CLS:D=0:GOTO 160
280 NEXT
290 IF D<>C-1 THEN PRINT"Você d
eve digitar";C;"palavras ";I$;"
 - repita":GOSUB 950:CLS:D=0:GO
TO 160
300 Z$=""
310 Z$=STRING$(L,"*")
320 CLS:PRINTA$;TAB(25)B$:PRINT
TA;"pontos";TAB(25);TB;"pontos"
330 LOCATE 7,3:PRINT"Valores do
s caracteres"
340 Q$="":FOR N=1 TO 28:READ G$
:Q$=Q$+G$+STRING$(4+(N/5=INT(N/
5)),32):NEXT:PRINTQ$:RESTORE
350 LOCATE 0,11:PRINT"A ";R$;"
contém";L;"letras":PRINTZ$
360 LOCATE 0,21:PRINTSPC(77):LO
CATE 0,21:PRINT"XX-Adivinha let
ra    ZZ-Adivinha frase":INPUT"
          A-Z Compra letra ";D$
900 DATA A-20,B-08,C-12,Ç-06,D-
20,E-20,F-12,G-08,H-08,I-16,J-0
8,K-02,L-10,M-12,N-12
910 DATA O-20,P-08,Q-08,R-12,S-
20,T-12,U-16,V-08,W-02,X-08,Y-0
2,Z-06,_-20
```

```
10 R$ = "PALAVRA":W = 14:F = 1:
TA = 200:TB = TA
40  HOME : INPUT "NOME DO JOGAD
OR 1 (MAX 9 LET) ";A$
50  PRINT : INPUT "NOME DO JOGA
DOR 2 (MAX 9 LET) ";B$
60  IF  LEN (A$) > 9 OR  LEN (B
$) > 9 THEN 40
70  HOME : PRINT "NIVEL DE DIFI
CULDADE": INPUT "(NUMERO DE PAL
AVRAS 1-9)? ";C$
90  IF C$ < "1" OR C$ > "9" THE
N 70
100 C =  VAL (C$)
110  PRINT : INPUT "NUMERO DE J
OGADAS (1-9)? ";T$
130  IF T$ < "1" OR T$ > "9" TH
EN 110
140 T =  VAL (T$)
150  IF C > 1 THEN J$ = "S":I$
 = "COM UM ESPACO ENTRE ELAS":R$
 = "FRASE"
160  HOME : PRINT A$", E A SUA
VEZ.": PRINT "DIGITE SUA FRASE
DE ";C;" PALAVRA";J$;"."
165  PRINT : PRINT "SE DESEJAR
VER AS LETRAS DIGITADAS, PRESSI
ONE <0>, SENAO PRESSIONE <1>.";

170  GET K$: IF K$ <  > "1" AND
 K$ <  > "0" THEN 170
180 K =  VAL (K$): PRINT : PRIN
T
185 S$ = " "
190  GET K$: IF K = 0 THEN  PRI
NT K$;
200  IF K$ =  CHR$ (8) AND  LEN
 (S$) > 1 THEN S$ =  LEFT$ (S$,
 LEN (S$) - 1): GOTO 190
210 S$ = S$ + K$: IF K$ <  >  C
HR$ (13) THEN 190
220 S$ =  MID$ (S$,2, LEN (S$)
 - 2):L =  LEN (S$)
230  IF L = 0 THEN  PRINT : PRI
NT "ENTRADA ILEGAL - REPITA"; C
HR$ (7);: GOSUB 950: HOME : GOT
O 160
240  IF L > 40 THEN  PRINT : PR
INT "ENTRADA MUITO LONGA - REPI
TA"; CHR$ (7);: GOSUB 950: HOME
 : GOTO 160
250  FOR N = 1 TO L: IF  MID$ (
S$,N,1) =  CHR$ (32) THEN D = D
 + 1: GOTO 270
260  IF  MID$ (S$,N,1) < "A" OR
  MID$ (S$,N,1) > "Z" THEN  PRI
NT : PRINT "CARATER ILEGAL - RE
PITA": GOSUB 950: HOME :D = 0:
GOTO 160
270  IF C = 1 AND D = 1 THEN  P
RINT : PRINT "ESPACOS NAO SAO P
ERMITIDOS EM UMA      UNICA PAL
AVRA! - REPITA": GOSUB 950: HOM
E :D = 0: GOTO 160
280  NEXT
290  IF D <  > C - 1 THEN  PRIN
T : PRINT "VOCE DEVE ENTRAR ";C
;" PALAVRAS ";I$;" - REPITA": G
OSUB 950: HOME :D = 0: GOTO 160

300 Z$ = " "
310  FOR N = 1 TO L:Z$ = Z$ + "
*": NEXT
320  HOME : PRINT A$; TAB( 25);
B$: PRINT TA;" PONTOS"; TAB( 25
);TB;" PONTOS"
330  VTAB 4: HTAB 8: PRINT "VAL
ORES DOS CARACTERES";
340 Q$ = " ": FOR N = 1 TO 28:
READ G$:Q$ = Q$ + G$ + "    ":
NEXT : HTAB 40: PRINT Q$: RESTO
RE
350  VTAB 12: PRINT "A ";R$;" C
ONTEM ";L;" LETRAS";: HTAB 40:
PRINT Z$
360  VTAB 22: CALL  - 958: VTAB
 22: PRINT "XX-ADIVINHA LETRA
  ZZ-ADIVINHA FRASE": INPUT "
       A-Z COMPRA A LETRA ";D$
900  DATA   A-20,B-08,C-12,D-20
,E-20,F-12,G-08,H-08
910  DATA   I-16,J-08,K-02,L-10
,M-12,N-12,O-20,P-08,Q-08,R-12,
S-20
920  DATA  T-12,U-16,V-08,W-02,
X-04,Y-02,"    ",Z-06,_-20
```

É possível transformar o programa em um jogo de palavras cruzadas?

Embora nosso programa não possa ser transformado num jogo desse tipo, ele certamente mostra o caminho para que o leitor elabore seu próprio programa.

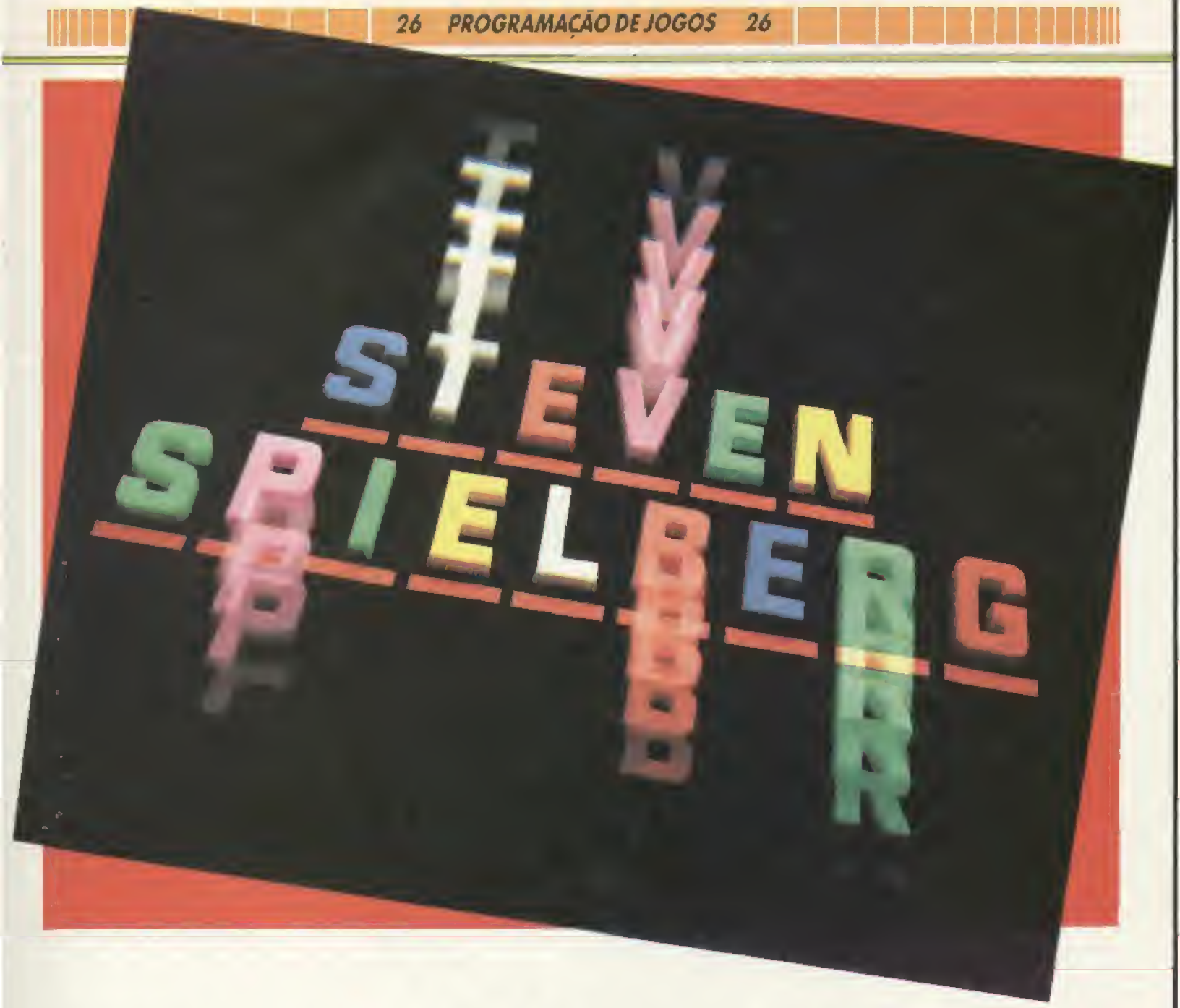
Num jogo de palavras cruzadas, o jogador também deve descobrir quais são as letras de palavras "escondidas". A diferença é que no nosso jogo precisamos descobrir os caracteres de um cordão, enquanto no de palavras cruzadas, buscamos elementos de uma matriz alfanumérica.

O tipo de pista que se dá ao jogador também é diferente. As instruções de um jogo de palavras cruzadas podem ocupar um espaço tão grande quanto os textos de um pequeno jogo de aventura. A disposição espacial das letras na tela também é muito importante. No programa, devemos correlacionar as coordenadas X e Y da tela aos números de linha da matriz de letras.

O programa é muito parecido para todos os microcomputadores, já que não há gráficos que requeiram comandos especiais para montá-los. Apenas o início difere um pouco em alguns casos. No MSX e no TRS-Color é preciso reservar espaço para as variáveis alfanuméricas que serão usadas.

A linha 10 define todas as variáveis necessárias para o jogo. No TRS-Color, os **PEEK** da linha 15 são utilizados mais tarde para evitar que algumas letras apareçam na tela. As linhas 20 e 30 do Spectrum definem um UDG em branco para a tabela de letras.

As linhas 40 e 70 dão as mensagens iniciais do jogo. O nome dos jogadores é chamado pelas linhas 40 e 50. A linha 60 verifica se eles não são muito compridos para o espaço disponível na tela.



O número de palavras da frase é escolhido na linha 70.

As linhas 80 a 100 fazem verificações com a finalidade de se certificar de que o número de palavras por frase está dentro dos limites do programa.

Esse número é representado pela variável **C$**. A linha 80 (quando existente) verifica se a entrada é de apenas um caractere; a linha 90 checa se o valor digitado está entre os números 1 e 9. A linha 100 converte esse valor para uma variável numérica.

As linhas 110 a 140 estão relacionadas ao número de jogadas escolhido. A linha 110 apresenta a mensagem e obtém a resposta (variável **T$**). As linhas 120 e 130 são similares às linhas 80 e 90. A linha 140 converte o valor para uma variável numérica.

Se a frase for constituída de mais de uma palavra, a linha 150 diz ao jogador que coloque apenas um espaço entre cada palavra. **R$** é definida como "FRASE" para uso posterior.

O programa passa, então, para a rotina de entrada da frase que um dos jogadores deverá adivinhar. Ela vai da linha 160 até a 220 e fornece instruções para o jogador que vai digitar a frase. Se ele selecionar 0, a frase aparecerá na tela; caso contrário, ela permanecerá invisível. A frase é armazenada na variável **S$**.

As linhas 230 a 290 conferem a frase para verificar se está de acordo com as regras. Se ela não tiver nenhum caractere, isto é, se a tecla <**ENTER**> foi pressionada antes de se digitar qualquer letra, a linha 230 anuncia uma entrada ilegal e pede que se repita a operação. A linha 240 verifica o tamanho da frase e a 250 o número de espaços (que deve ser um a menos que o número de palavras).

As linhas 260 e 270 procuram caracteres ilegais. Observe que apenas letras maiúsculas serão aceitas, sem acento. No MSX, pode-se usar o "Ç".

As linhas 300 e 310 são encarregadas de definir a variável **Z$**, que contém a seqüência de asteriscos de número igual ao de letras de **S$**.

A rotina final, das linhas 320 a 360, monta o que falta da tela principal, lendo a tabela de valores de linhas **DATA** e colocando a tabela na posição adequada. A seqüência de asteriscos também é posicionada. A linha 360 apresenta as opções para o jogador que vai adivinhar a frase.

APERFEIÇOE SEU BANCO DE DADOS

Apresentamos aqui novas rotinas para o seu banco de dados. Elas tornarão o programa mais eficiente, permitindo-lhe organizar melhor as informações que desejar.

- ADIÇÃO DE NOVAS OPÇÕES
- COMO MELHORAR AS ROTINAS JÁ EXISTENTES
- IMPRESSÃO CONTÍNUA
- UMA NOVA ORGANIZAÇÃO DO ARQUIVO
- BUSCA DE MÚLTIPLAS INFORMAÇÕES
- O USO DO CONTROLADOR DE DISCOS

O programa de banco de dados que desenvolvemos nos artigos das páginas 68 e 81 é muito útil para armazenar informações. Não importa que sejam detalhes de seu passatempo predileto, resultados de uma pesquisa, endereços de amigos e clientes ou qualquer outro tipo de dado. Devidamente arquivados, eles poderão ser consultados, corrigidos, alterados, apagados e mesmo impressos em papel.

Como qualquer outro programa de banco de dados, o nosso também não é perfeito para todas as aplicações possíveis. Para que você possa adaptá-lo da melhor maneira às suas necessidades, apresentamos algumas novas rotinas assim como sugestões para o aperfeiçoamento das já existentes. As rotinas, diferentes para cada computador, serão examinadas separadamente.

S

Para os usuários do Sinclair Spectrum, selecionamos duas rotinas: uma para a entrada contínua de registros e outra para impressão contínua.

ENTRADA CONTÍNUA

Estas linhas lhe permitirão dar entrada aos registros continuamente, sem precisar retornar ao menu. Elas impedem também que um cordão nulo — que provocaria a volta ao menu — seja aceito. Pressione <ENTER> quando completar a entrada dos dados.

```
2000 CLS : LET C=V
2110 FOR N=V TO A: PRINT  INVER
SE V;AT V+N*2,0;N$(N);AT V+N*2,
12; FLASH V;"?": INPUT "(ate ";
A(N);" caracteres)", LINE A$(C,
B(N)+V TO B(N+V)): IF N=V AND A
$(C,B(N)+V)=CHR$ 32 THEN  RETUR
N
2115 PRINT AT V+N*2,12;A$(C,B(N
)+V TO B(N+V)): NEXT N
2120 FOR F=V TO 150: NEXT F: IF
 C=V THEN  GOTO 2000
2140 IF A$(C)>=A$(C-V) THEN  GO
TO 2000
2150 LET X$=A$(C): LET A$(C)=A$
(C-V): LET A$(C-V)=X$: LET C=C-
V: IF C=V THEN  GOTO 2000
```

IMPRESSÃO CONTÍNUA

Como está, o programa possibilita a impressão apenas do registro em exame. Para imprimir a lista inteira, será necessário passar por todo o arquivo e comandar a impressão. Esta nova rotina faz o trabalho por você, iniciando pelo registro que está sendo exibido e imprimindo todos os seguintes até o fim da lista. Se quiser imprimir todos os registros, certifique-se de que está no registro 1, antes de iniciar o processo. Para interromper a listagem, pressione qualquer tecla.

```
 120 LET OP=1
3015 IF D-V=R THEN  LET D=D-V:
IF OP=6 THEN  LET OP=1
3020 IF A$(D,V)=CHR$ 32 THEN  L
ET D=D-V: IF OP=6 THEN  LET OP=
1
3085 IF OP=6 THEN  LET D=D+V: G
OTO 3010
4060 IF D>R THEN  LET D=PM: IF
OP=6 THEN  LET OP=1
4080 IF A$(D,V)=CHR$ 32 THEN  L
ET D=PM: IF OP=6 THEN  LET OP=1
4165 IF OP=6 THEN  LET MO=V: LE
T D=D+MO: GOTO 4060
9502 IF OP=6 AND INKEY$="" THEN
  COPY : RETURN
9505 PRINT  INVERSE V;AT 19,U;"
 impressao (C)ontinua";TAB 31;"
"
9585 IF V$="C" THEN  COPY : LET
 OP=6
```

T

Apresentamos quatro rotinas para o seu TRS-Color: impressão contínua, reordenação do arquivo, procura por múltiplos campos e adaptação para acionadores de disco.

IMPRESSÃO CONTÍNUA

Com esta rotina, você poderá imprimir todos os registros de uma só vez. Selecione a opção de impressão e escolha entre impressão contínua ou registro único. A impressão começa no registro que está sendo exibido na tela e vai até o fim da lista.

Se você quiser imprimir todo o arquivo, não se esqueça que deve começar com o registro 1.

```
1050 PRINT @385,"NUMERO DE CAMP
OS (1-8) ?";
1060 IN$=INKEY$:IF IN$<"1" OR I
N$>"8" THEN 1060
1070 A=VAL(IN$):DIM A(A),N$(A)
5070 IF D>NR AND G=1 THEN G=0:C
H=-1:CP=0 ELSE IF D>NR THEN CP=
0:GOTO 5230
5105 IF CP=1 GOSUB 10040:GOTO 5
160
5210 GOSUB 10000:GOTO 5160
6025 IF CP=1 GOSUB 10040:GOTO 6
080
6130 GOSUB 10000: IF CP=1 THEN
6080 ELSE 6030
6140 IF D>NR THEN D=1:CP=0
10000 PRINT @449," AJUSTE A IMP
RESSORA     cONT";STRING$(36,32
);
10010 IF INKEY$<>"C" THEN 10010
10020 PRINT @448,"cONTINUA OU s
OMENTE UM REGISTRO?";
10030 IN$=INKEY$:IF IN$<>"C" AN
D IN$<>"S" THEN 10030
10035 IF IN$="C" THEN CP=1
```

REORDENAÇÃO DO ARQUIVO

O programa original faz a ordenação dos registros sempre pelo mesmo campo (o primeiro). A nova alternativa, que aparecerá com o número 8 no menu, permite que o campo-chave para a ordenação seja mudado. Depois da mudança, o arquivo é reordenado e o campo-chave passa a ser o primeiro da lista.

```
105 PRINT @388,"8-REORGANIZAR C
AMPOS"
120 IN$=INKEY$:IF IN$<"1" OR IN
$>"8" THEN 120
150 ON IN GOSUB 1000,2000,6000,
5000,7000,8000,9000,11000
11000 IF A=1 THEN PRINT" NAO PO
SSO REORGANIZAR 1 CAMPO!":FOR K
=1 TO 5000:NEXT:RETURN
11010 PRINT "NUM.DO CAMPO","NOM
E"
11020 FOR N=1 TO A:PRINT N,N$(N
):NEXT
11030 PRINT:PRINT" DIGITE O NUM
ERO DO NOVO CAMPO   CHAVE (2 A"
;A;")";:INPUT NC
11040 NC=INT(NC):IF NC<2 OR NC>
A THEN CLS:GOTO 11010
11050 CLS:PRINT" TROCANDO E ORD
ENANDO CAMPOS"
11060 T$=N$(1):N$(1)=N$(NC):N$(
```

```
NC)=T$:T=A(1):A(1)=A(NC):A(NC)=
T
11070 FOR N=1 TO NR:T$=A$(N,1):
A$(N,1)=A$(N,NC):A$(N,NC)=T$:NE
XT
11080 FOR N=1 TO NR-1:K=N
11090 FOR J=N+1 TO NR
11100 IF A$(J,1)<A$(K,1) THEN K
=J
11110 NEXT:IF N<>K THEN FOR C=1
 TO A:T$=A$(K,C):A$(K,C)=A$(N,C
)=T$:NEXT
11120 NEXT:RETURN
```

PROCURA POR MÚLTIPLOS CAMPOS

Esta nova opção permite que você busque mais de uma informação por vez. Quando se seleciona a rotina de busca a partir do menu principal, deve-se responder, para cada campo, o que será procurado. Você pode procurar por quantos campos desejar. Se um deles não lhe interessar, simplesmente tecle <ENTER> e vá em frente.

Em seguida, o computador perguntará se os dados especificados deverão estar presentes em todos os registros ou não. Se sua resposta for SIM, apenas os registros que contenham todos os dados especificados serão mostrados. Caso você responda NÃO, qualquer registro que contenha pelo menos um dos dados será exibido. Se, por exemplo, você solicitar o nome MARIA para o campo 1 e CAMPINAS para o campo 3, apenas pessoas que se chamem Maria E residam em Campinas serão apresentadas caso você tenha respondido SIM à pergunta anterior. Caso contrário, todas as pessoas que se chamem Maria **OU** residam em Campinas serão listadas.

```
5000 PRINT @B,B$;"opcao";B$;"de
";B$;"procura";B$
5010 BT=0:PRINT:FOR N=1 TO A:PR
INT "PROCURA PELO QUE NO CAMPO"
:PRINT N;", ";N$(N);" ?"
5020 LINEINPUT S$(N):IF S$(N)<>
"" THEN BT=BT+1
5025 NEXT:IF BT=0 THEN RETURN
5027 IF BT<2 THEN BT=0:GOTO 506
0
5030 PRINT:PRINT" TODOS OS DADO
S DEVEM ESTAR SI-  MULTANEAMENT
E NO MESMO REGISTRO (S/N) ?";
5040 IN$=INKEY$:IF IN$<>"S" AND
 IN$<>"N" THEN 5040
5050 CLS:BT=0:IF IN$="S" THEN B
T=1
5090 FOR Z=1 TO A:PS=INSTR(A$(D
,Z),S$(Z)):IF PS>0 AND BT=0 AND
 S$(Z)<>"" THEN Z=A:NEXT:GOTO 5
100
5093 IF PS=0 AND BT=1 THEN Z=A:
NEXT:D=D+CH:GOTO 5070
5096 NEXT:IF BT=0 THEN D=D+CH:G
OTO 5070
5230 CLS 2:PRINT "  NENHUM REGI
STRO COM";:IF BT=1 THEN PRINT @
21," TODOS OS",ELSE PRINT @21,"
 ALGUM DOS"
5231 PRINT "  DADOS FOI ENCONTR
ADO";
5235 FOR Z=1 TO A:IF S$(Z)="" T
HEN 5245
5240 PRINT @96+Z*32,N$(Z);:PRIN
T @107+Z*32,S$(Z);
5245 NEXT:CP=0
```

O USO DO ACIONADOR DE DISCOS

As linhas que se seguem fazem com que o programa trabalhe com o auxílio de um acionador de discos. Se você usa um gravador cassete, não as digite, pois as rotinas de carregamento e gravação serão alteradas.

Antes de introduzir as novas linhas,

você deverá apagar as linhas 8060 a 8070, 8150 a 8200 e 7090 a 7140. A maneira mais fácil de fazê-lo é digitar **DEL nº inicial nº final**. Por exemplo: **DEL 8060-8070**.

Para transferir dados da fita cassete para o acionador de discos, carregue o programa e faça as alterações da rotina de gravação (linhas até 7080). Carregue os dados e grave-os em disco. Apague, então, as linhas de 8060 em diante (só as especificadas antes). Por fim, digite as novas linhas da rotina de carregamento de dados.

```
80 PRINT@292,"5-SALVAR ARQUIVO"
90 PRINT @324,"6-CARREGAR ARQUI
VO"
1130 NEXT:R=INT(9000/(5+5*A))-1
:PRINT "NUMERO MAXIMO DE REGIST
ROS=";R
1140 DIM A$(R,A):FOR I=1 TO 200
0:NEXT:RETURN
7000 CLS:PRINT" CERTIFIQUE-SE D
E QUE O DRIVE    ESTA LIGADO E
O DISCO INSERIDO  E PRESSIONE <
ENTER>
7010 IF INKEY$<>CHR$(13) THEN 7
010
7020 PRINT:PRINT" QUAL O NOME D
O ARQUIVO ?";:LINEINPUT FI$
7030 IF LEFT$(FI$,1)<"A" OR LEF
T$(FI$,1)>"Z" THEN 7020
7040 OPEN "O",#1,FI$+"/DAT":CLS
6:PRINT @232," GRAVANDO ";FI$;
7050 WRITE #1,R,A,NR
7060 FOR N=1 TO A:WRITE #1,N$(N
),A(N):NEXT
7070 FOR C=1 TO NR:FOR N=1 TO A
:WRITE #1,A$(C,N):NEXT N,C
7080 CLOSE#1:RETURN
8030 PRINT @65,"SELECIONE O DIS
CO, PRESSIONE    <ENTER>"
8040 IF INKEY$<>CHR$(13) THEN 8
040
8050 IF R>0 THEN RUN 9210
8080 PRINT:PRINT" QUAL O NOME D
O ARQUIVO ?";:LINEINPUT FI$
8090 IF LEFT$(FI$,1)<"A" OR LEF
T$(FI$,1)>"Z" THEN 8080
8100 OPEN "I",#1,FI$+"/DAT"
8105 INPUT#1,R,A,NR
8110 DIM A(A),N$(A),A$(R,A)
8120 FOR N=1 TO A:INPUT#1,N$(N)
,A(N):NEXT
8130 FOR C=1 TO NR:FOR N=1 TO A
:INPUT#1,A$(C,N):NEXT N,C
8140 CLOSE#1:RETURN
```

IMPRESSÃO CONTÍNUA

Apresentamos três rotinas a mais para o MSX. Com elas você poderá fazer uma listagem contínua dos seus dados, reorganizar o arquivo e buscar várias informações ao mesmo tempo.

A rotina para impressão contínua permite que vários registros do seu arquivo sejam listados de uma só vez. O registro a partir do qual você quer começar a listagem deve estar sendo exibido na tela. Selecione, então, a opção [I]mprimir e, a seguir, a opção [C]ontínua. Todos os registros a partir deste serão listados.

Com a opção [R]egistro você pode imprimir apenas aquele que está sendo mostrado. Não se esqueça de que, para listar todo o arquivo, você deve ter o registro nº 1 da tela.

```
5070 IFD>NRANDG=1THEN G=0:CH=-1
:CP=0 ELSEIFD>NRTHEN CP=0 :GOTO
 5230
5105 IF CP=1 THEN GOSUB 10040:G
OTO 6080
5210 GOSUB10000:GOTO5160
6025 IF CP=1 THEN GOSUB 10040:G
OTO 6080
6130 GOSUB10000:IF CP=1 THEN 60
80 ELSE 6020
6140 IFD>NRTHEND=1:CP=0
10030 PRINT:LOCATE 1:PRINT"IMPR
ESSAO [C]ONTINUA OU [R]EGISTRO?
 ";
10035 IN$=INKEY$:IFIN$<>"C"ANDI
N$<>"R"THEN10035
10037 IFIN$="C" THEN CP=1
```

REORGANIZAÇÃO DO ARQUIVO

Esta nova opção permite que se troque o campo-chave para a ordenação do arquivo. O programa original tomava automaticamente o primeiro campo para fazer a ordenação alfanumérica dos registros. Com a possibilidade de trocar esse campo, ficará fácil, por exemplo, organizar por autor um arquivo de livros que estava indexado por título da obra. A operação pode ser revertida a qualquer momento. O novo campo-chave aparecerá em primeiro lugar na listagem dos campos.

```
105 LOCATE 9,19:PRINT"8:-REORGA
NIZAR O ARQUIVO"
110 LOCATE 14,22:PRINT"OPÇÃO: "
;
120 IN$=INKEY$:IF IN$<"1"ORIN$>
"8"THEN120
150 ON IN GOSUB 1000,2000,6000,
5000,7000,8000,9000,11000
11000 IF A=1 THEN PRINT:PRINT"I
MPOSSIVEL REORGANIZAR APENAS 1
CAMPO!":FOR K=1 TO 5000:NEXT:RE
TURN
11010 PRINT:PRINT"CAMPO N.","NO
ME":PRINT
11020 FOR N=1 TO A:PRINTN,N$(N)
:NEXT
11030 PRINT:PRINT"DIGITE O NUME
RO DO NOVO CAMPO CHAVE":PRINT"(
 2 A";A;")";:INPUT NC
11040 IF NC<2 OR NC>A THEN CLS:
GOTO 10010
11050 CLS:LOCATE7,18:PRINT"REOR
GANIZANDO E REORDENANDO":PRINT:
PRINTTAB(15)"O ARQUIVO"
11060 SWAP N$(1),N$(NC):SWAP A(
1),A(NC)
11070 FOR N=1 TO NR:SWAP A$(N,1
),A$(N,NC):NEXT
11080 FOR N=1 TO NR-1:K=N
11090 FOR J=N+1 TO NR
11100 IF A$(J,1)<A$(K,1) THEN K
=J
11110 NEXT:IF N<>K THEN FOR C=1
 TO A:SWAP A$(K,C) A$(N,C):NEXT
11120 NEXT:RETURN
```

Como modificar o programa para trabalhar com arquivos em disco?

Os micros das linhas TRS-Color e MSX podem usar discos flexíveis como meio magnético de armazenamento de dados. Os programas de INPUT, contudo, destinam-se a equipamentos que utilizam fitas cassete.

Se quisermos usar em discos nosso programa para gerar e controlar arquivos, precisaremos modificar as seções que abrem estes arquivos e as que gravam e recuperam dados deles. O procedimento é necessário porque o programa não modifica registros isolados no arquivo gravado, e sim na memória do computador. Um conjunto de registros é criado na memória e, depois, gravado na fita. Quando queremos alterar alguma coisa, todo o bloco é trazido para a memória, onde o programa faz a edição. Depois que o processo se completa, gravamos o bloco em fita.

Para fazer essas modificações, o usuário precisa, naturalmente, conhecer os comandos correspondentes para manipulação de arquivos em disco. Estes deverão ser do tipo seqüencial, como os da fita.

BUSCA DE MÚLTIPLAS INFORMAÇÕES

Com a rotina dada a seguir você terá a alternativa de procurar por mais de uma informação de cada vez. Quando selecionar a opção de busca, você precisará especificar as informações que deseja em cada campo do registro. Se um determinado campo não lhe interessa, simplesmente tecle <**RETURN**>. Depois, o computador perguntará se os dados especificados devem estar simultaneamente em cada registro ou não. A resposta afirmativa corresponde a um E na

busca — ou seja, o registro deve conter isto E isto E aquilo. O contrário corresponde a um **OU**. Vejamos um exemplo. Você procura por MACHADO DE ASSIS no campo AUTOR e ROMANCE no campo GÊNERO LITERÁRIO. Se sua resposta para a pergunta anterior for SIM, apenas os romances de Machado de Assis serão listados. Se sua resposta for NÃO, todos os livros de Machado e todos os romances serão listados.

```
5000 PRINT"BUSCA DE INFORMAÇÕES
"
5010 BT=0:PRINT:FOR N=1 TO A:PR
INT"PROCURAR O QUE EM ";N$(N);"
? ";:S$(N)=""
5020 LINEINPUTS$(N):IF S$(N)<>"
" THEN BT=BT+1
5025 NEXT:IF BT=0 THEN RETURN
5027 IF BT=1 THEN BT=0:GOTO 506
0
5030 PRINT:PRINT:PRINT"TODOS OS
 DADOS DEVEM ESTAR PRESENTES
 SIMULTANEAMENTE? ";
5040 IN$=INKEY$:IF IN$<>"S" AND
 IN$<>"N" THEN 5040
5050 CLS:BT=0:IF IN$="S" THEN B
T=1
5090 FOR Z=1 TO A:PS=INSTR(A$(D
,Z),S$(Z)):IF PS>0 AND BT=0 AND
 S$(Z)<>"" THEN Z=A:NEXT:GOTO 5
100
5093 IF PS=0 AND BT=1 THEN Z=A:
NEXT:D=D+CH:GOTO 5070
5096 NEXT:IF BT=0 THEN D=D+CH:G
OTO 5070
5230 CLS:LOCATE 5,10:PRINT"NENH
UM REGISTRO COM ";:IF BT=1 THEN
 PRINT"TODOS OS" ELSE PRINT"ALG
UM DOS"
5235 PRINT"FOI ENCONTRADO!":CP=
0
```

Para o Apple II, temos quatro rotinas. As três primeiras possibilitam a impressão contínua de registros, a reorganização do seu arquivo e a busca de várias informações ao mesmo tempo. A última delas melhora rotinas já existentes no programa, tornando-o, desse modo, mais completo.

IMPRESSÃO CONTÍNUA

Esta pequena rotina permite que os registros sejam listados em seqüência, e não apenas um de cada vez. Deixe na tela o primeiro registro a ser impresso e escolha a opção de impressão. Em seguida, tecle **C** para a impressão contínua. Todos os registros, a partir do que está na tela, serão listados. Para a listagem de todo o arquivo, deixe o registro 1 na tela e proceda como foi explicado.

```
10017  VTAB 23: HTAB 5: CALL -
 958: PRINT "IMPRESSAO ";: INVE
RSE : PRINT "C";: NORMAL : PRIN
T "ONTINUA OU ";: INVERSE : PRI
NT "R";: NORMAL : PRINT "EGISTR
O ?";
10018  GET IN$: IF IN$ < > "C"
 AND IN$ < > "R" THEN 10018
10025  IF IN$ = "C" THEN E = D:
 FOR D = E TO NR
10052  IF IN$ = "C" THEN  NEXT
:D = E
```

REORGANIZAÇÃO DO ARQUIVO

O programa original usava o primeiro campo do arquivo como campo-chave para a ordenação dos registros. Com esta rotina, você poderá trocar o campo. Selecione a opção 8 a partir do menu principal e especifique qual o novo campo-chave. O arquivo será, então, reordenado de acordo com a escolha feita. O novo campo-chave aparecerá no topo da lista de campos.

```
105  PRINT : HTAB 10: PRINT "8:
-REORGANIZAR O ARQUIVO"
110  VTAB 23: HTAB 15: PRINT "O
PCAO: ";
130  IF IN$ < "1" OR IN$ > "8"
THEN 110
150  ON IN GOSUB 1000,2000,6000
,5000,7000,8000,6520,11500


11500  IF A = 1 THEN  VTAB 10:
HTAB 9: PRINT "IMPOSSIVEL REORG
ANIZAR": PRINT  TAB( 13)"APENAS
 1 CAMPO!": FOR K = 1 TO 3000:
NEXT : RETURN
11510  PRINT : PRINT "CAMPO NO.
","NOME"
11520  FOR Z = 1 TO A: PRINT Z,
N$(Z): NEXT
11530  PRINT : PRINT "DIGITE O
NUMERO DO NOVO CAMPO CHAVE": PR
INT "(DE 2 A ";A;")";: INPUT NC
%
11540  IF (NC% < 2) OR (NC% > A
) THEN  HOME : GOTO 11510
11550  HOME : VTAB 10: HTAB 8:
PRINT "AGUARDE A  REORGANIZACAO
": PRINT  TAB( 9)"E ORDENACAO D
O ARQUIVO"
11560 T$ = N$(1):N$(1) = N$(NC%
):N$(NC%) = T$:T = A(1):A(1) =
A(NC%):A(NC%) = T
11570  FOR N = 1 TO NR:T$ = A$(
N,1):A$(N,1) = A$(N,NC%):A$(N,N
C%) = T$: NEXT
11580  FOR N = 1 TO NR - 1:K =
N
11590  FOR J = N + 1 TO NR
11600  IF A$(J,1) < A$(K,1) THE
N K = J
11610  NEXT : IF N <  > K THEN
 FOR C = 1 TO A:T$ = A$(K,C):A$
(K,C) = A$(N,C):A$(N,C) = T$: N
EXT
11620  NEXT : RETURN
```

BUSCA DE MÚLTIPLAS INFORMAÇÕES

Esta é uma rotina destinada a agilizar a consulta ao seu arquivo de dados, permitindo-lhe procurar por mais de uma informação de cada vez. Quando você solicitar a opção de busca, precisará especificar as informações que deseja em cada campo. Se um campo não lhe interessa, simplesmente tecle <ENTER>. Depois, o computador perguntará se os dados especificados devem estar simultaneamente em cada registro ou não. A resposta afirmativa fará com que apenas os registros que contêm todos os dados especificados sejam apresentados. A resposta negativa levará todo registro que contenha ao menos um dos dados a ser listado.

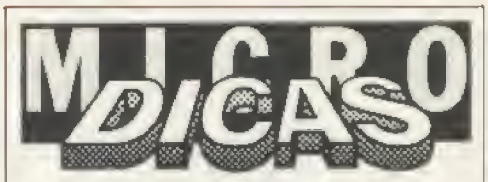
```
5000 DD = 1:BT = 0: PRINT : FOR
 X = 1 TO A: PRINT : PRINT "PRO
CURAR O QUE EM ";N$(X);
5010  INPUT S$(X): IF S$(X) <
> "" THEN BT = BT + 1
5020  NEXT : IF BT = 0 THEN  RE
TURN
5025  IF BT = 1 THEN BT = 0: GO
TO 5040
5030  VTAB 21: HTAB 1: PRINT "T
ODOS OS DADOS DEVEM ESTAR PRESE
NTES     EM CADA REGISTRO? (S/N)
";
5035  GET IN$: IF IN$ <  > "S"
AND IN$ <  > "N" THEN 5035
5037  IF IN$ = "N" THEN BT = 0
5040  VTAB 23: HTAB 13: PRINT "
PROCURANDO...";
5050  FOR XX = DD TO NR
5052  FOR Z = 1 TO A: IF S$(Z)
= "" THEN 5060
5054 PS = (S$(Z) =  LEFT$ (A$(X
X,Z), LEN (S$(Z)))): IF PS AND
( NOT BT) THEN Z = A: NEXT : GO
TO 5065
5056  IF ( NOT PS) AND BT THEN
Z = A: NEXT : NEXT : GOTO 5070
5060  NEXT : IF BT = 0 THEN  NE
XT : GOTO 5070
5065 FL = 1:D = XX: GOSUB 6020:
 GOTO 5200
5070  IF FL = 0 THEN  VTAB 21:
HTAB 1: CALL  - 958: PRINT "NAO
 ENCONTREI NENHUM REGISTRO COM
OS     DADOS SOLICITADOS!": FOR
 X = 1 TO 5000: NEXT : GOTO 509
0
5090 FL = 0: RETURN
5200  IF XX = NR THEN 5070
5210  VTAB 15: PRINT "CONTINUO
PROCURANDO? (S/N) ";: GET IN$
```

MELHORE AS ROTINAS EXISTENTES

Estas linhas tornam seu programa de banco de dados mais completo. Com elas você poderá especificar o *slot* e a unidade de disco que usará para a gravação e o carregamento dos dados. A opção padrão é slot 6 e *drive* 1. Um segundo drive poderá ser acionado diretamente do programa.

Alteramos ainda a rotina de detecção de erros. Ela está mais explícita em relação à não existência de um arquivo especificado para a leitura de dados. E também mais equipada para prevenir que qualquer outro erro (a seleção de um drive que não existe, por exemplo) interrompa seu programa e provoque a perda de dados.

```
22 SS = 6:DR = 1
25  ONERR  GOTO 11000
1030  IF R > 0 THEN  CLEAR :D$
=  CHR$ (4):IN = 1:SS = 6:DR =
1: HOME : GOTO 150
1050  VTAB 3: HTAB 5: PRINT "NU
MERO DE CAMPOS (1-8): ";
1060  GET A$: IF A$ > "8" OR AS
 < "1" THEN 1060
1070  PRINT A$:A =  VAL (A$): D
IM A(A),N$(A)
3010  VTAB 23: CALL  - 958: PRI
```



NÃO PERCA SEUS DADOS

Às vezes, por descuido ou erro de digitação, o programa é interrompido por uma mensagem de erro. Se executarmos o programa de novo, usando **RUN**, todas as variáveis criadas pelo programa serão apagadas.

A saída para o problema está em evitar o comando **RUN** e voltar a executar o programa de outro modo.

Em geral, é possível encontrar um ponto onde o programa recomeça como se nada tivesse acontecido. Na maioria das vezes, este ponto corresponde à parte que cuida do menu. Para executar o programa novamente, bastará, então, anotar o número da linha e usar o comando **GOTO**.

```
NT "NUMERO DO CAMPO A SER MODIF
ICADO =>";
3020  GET CP$:CP =  VAL (CP$)
3030  IF CP > A OR CP < 1 THEN
 POKE 34,0: RETURN
7010  VTAB 15: HTAB 5: PRINT "S
LOT ";SS; CHR$ (8);: GET SS$: I
F SS$ =  CHR$ (13) THEN SS$ =
STR$ (SS)
7015  PRINT SS$:SS =  VAL (SS$)
: IF SS < 1 OR SS > 7 THEN 7010
7020  VTAB 17: HTAB 5: PRINT "D
RIVE ";DR; CHR$ (8);: GET DR$:
IF DR$ =  CHR$ (13) THEN DR$ =
 STR$ (DR)
7022  PRINT DR$:DR =  VAL (DR$)
: IF DR < 1 OR DR > 2 THEN 7020
7025  PRINT : PRINT D$;"OPEN";A
R$;",S";SS;",D";DR: PRINT D$;"D
ELETE";AR$
8020  HOME : CLEAR :D$ =  CHR$
(4):IN = 6:SS = 6:DR = 1
8050  VTAB 15: HTAB 5: PRINT "S
LOT ";SS; CHR$ (8);: GET SS$: I
F SS$ =  CHR$ (13) THEN SS$ =
STR$ (SS)
8055  PRINT SS$:SS =  VAL (SS$)
: IF SS < 1 OR SS > 7 THEN 8050
8060  VTAB 17: HTAB 5: PRINT "D
RIVE ";DR; CHR$ (8);: GET DR$:
IF DR$ =  CHR$ (13) THEN DR$ =
 STR$ (DR)
8065  PRINT DR$:DR =  VAL (DR$)
: IF DR < 1 OR DR > 2 THEN 8060
8070  PRINT : PRINT D$;"OPEN";A
R$;",S";SS;",D";DR
11000  IF  PEEK (222) <  > 5 TH
EN  VTAB 24: HTAB 13: PRINT "HO
UVE UM ERRO!";: FOR I = 1 TO 10
00: NEXT : GOTO 30
11010  PRINT D$;"CLOSE"
11020  VTAB 24: HTAB 8: PRINT "
ESTE ARQUIVO NAO EXISTE!";: FOR
 I = 1 TO 1000: NEXT : HOME : G
OTO 150
```

APPLE E TK-2000: EFEITOS SONOROS

O comando SOUND do TK-2000 tem certas limitações. Já o Apple não possui um comando BASIC para produzir sons. Veja como usar código de máquina para criar efeitos sonoros.

Os usuários do Apple ressentem-se, sem dúvida, da ausência de um comando em BASIC para produzir sons. Essa limitação pode ser contornada com o uso de rotinas em código de máquina.

Estalidos ou "cliques" produzidos no alto-falante constituem o máximo que se pode obter com um programa BASIC, no Apple, em matéria de som. Para isso (tanto no Apple quanto no TK-2000), basta que façamos referência à posição de memória -16336 por meio de um comando **PEEK**:

```
X = PEEK(-16336)
```

Quando testamos o valor desse endereço especial, o cone do alto-falante se move — para dentro e para fora — uma vez. Se quisermos produzir notas musicais, precisaremos imprimir a esse movimento uma velocidade que somente programas em linguagem de máquina permitem. Controlando a velocidade com que o cone do alto-falante se move para frente e para trás, poderemos controlar também a freqüência da nota musical ou do ruído emitido. Quanto mais rápido for o movimento do cone, mais agudo será o som.

COMO CONTROLAR O ALTO-FALANTE

A rotina em Assembly fornecida a seguir produz um som que vai se tornando cada vez mais agudo. Use nosso Assembler para montá-la na memória.

```
10 ORG 800
20 LDA #$00
30 STA $FF
40 LOOP LDA #$00
50 STA $C030
60 LDX $FF
70 PAUSE NOP
80 NOP
90 NOP
100 NOP
110 DEX
120 BNE PAUSE
130 DEC $FF
140 BEQ FIM
150 JMP LOOP
160 FIM RTS
170 END
```

Os usuários do TK-2000 podem recorrer ao míni-Assembler embutido no micro. O míni-Assembler, contudo, não aceita rótulos, seguindo outra listagem com os endereços já calculados. Ao contrário da listagem anterior, esta não é relocável na memória.

```
0320-   LDA   #$00
0322-   STA   $FF
0324-   LDA   #$00
0326-   STA   $C030
0329-   LDX   $FF
032B-   NOP
032C-   NOP
032D-   NOP
032E-   NOP
032F-   DEX
0330-   BNE   $032B
0332-   DEC   $FF
0334-   BEQ   $0339
0336-   JMP   $0324
0339-   RTS
```

Se você usar nosso Assembler para montar o programa, verá que ele emite

- COMO CONTROLAR O ALTO-FALANTE
- O USO DE CONTADORES
- O CONTROLE DA FREQÜÊNCIA DA NOTA
- COMO USAR LAÇOS VAZIOS PARA PRODUZIR PAUSAS DE DIVERSOS TAMANHOS
- UMA ROTINA DE SOM PARA O APPLE

mensagens de erro durante a montagem. Para corrigir esse defeito, modifique a linha 110: apague a segunda vírgula depois do **NOP** e acrescente uma vírgula após o número 234.

Qualquer referência à posição de memória -16336 em decimal, ou $C030 em hexadecimal, provocará um movimento no alto-falante. Essa referência pode ser feita por meio de comandos como **STA**, **INC** ou **DEC**. O importante, de fato, é controlar a freqüência com que isso ocorre no programa.

O USO DE CONTADORES

A posição de memória $FF — que fica na página zero — vai ser usada como contador. Para começar, colocamos nela o valor zero, pelos comandos **LDA #$00** e **STA $FF**. Depois, o comando **STA $C030** provoca um primeiro movimento do alto-falante. Note que, antes disso, foi colocado zero no registro A. Na realidade, o valor transferido de A para o endereço $C030 pelo comando **STA $C030** pode ser qualquer um: não faz a menor diferença.

A seguir, a instrução **LDX $FF** coloca no registro X o conteúdo da memória $00FF. Esse tipo de endereçamento é denominado *endereçamento direto em página zero*. Sua vantagem consiste em permitir a especificação de um endereço com apenas um byte. Como um byte pode conter um número entre 0 e 255, o endereço especificado deve ficar num dos primeiros 256 bytes da memória — a chamada *página um*.

As quatro instruções **NOP** (Nenhuma **OP**eração) utilizadas em seguida nada executam. Sua função é reduzir a velocidade do programa. A instrução **DEX** subtrai uma unidade do valor de X. Se o resultado da subtração não for igual a zero, a instrução **BNE** manda o processador de volta ao rótulo **PAUSE**. O laço que fica entre as linhas 70 e 110 é repetido, então, 256 vezes — lembre-se de que zero menos um é igual a 255 em linguagem de máquina.

Em seguida, a instrução **DEC $FF** subtrai uma unidade do conteúdo do endereço **$FF**. Se a operação resultar em zero, a instrução **BEQ FIM** envia o processador para o rótulo **FIM**, onde o comando **RTS** termina a rotina. Se a operação não resultar em zero, o programa segue seu curso natural e a instrução **JMP LOOP** envia o processador ao rótulo **LOOP**, para a produção de um novo movimento do alto-falante.

Assim, após a emissão do primeiro som, ocorre uma pausa equivalente a 256 voltas do laço. O conteúdo de $FF, que inicialmente é zero — equivalente a 256, em código de máquina —, controla a duração da pausa, diminuindo em uma unidade após esta. Enquanto não chegar a zero, um novo movimento do alto-falante será produzido. Como o conteúdo de $FF, que controla o tamanho da pausa, diminui a cada volta do laço entre as linhas 40 e 150, o tamanho da pausa entre dois movimentos do alto-falante também diminui. Dessa maneira, a freqüência do som produzido vai aumentando, ou seja, o som vai se tornando cada vez mais agudo.

No TK-2000, o som é produzido por intermédio do alto-falante da TV. Já no Apple um dispositivo sonoro que está

embutido no próprio computador emite o ruído — e a qualidade do som deixa a desejar.

A VEZ DO APPLE

A rotina de produção de ruídos que apresentamos a seguir pode ser utilizada a partir de programas BASIC, equivalendo ao comando **SOUND** dos micros da linha TK-2000.

Ela permite ao usuário do Apple produzir notas musicais em seus programas sem ter que programar uma rotina em linguagem de máquina para cada novo efeito sonoro.

```
10 ORG 800
20 LDY #$00
30 SOM LDX $385
40 INC $C030
50 PAUSET DEY
60 BNE PAUSES
70 DEC $384
80 BEQ FIM
90 PAUSES DEX
100 BNE PAUSET
110 JMP SOM
120 FIM RTS
```

Embora o TK-2000 dispense esse tipo de rotina, seus usuários podem querer montá-la para aprender ou, simplesmente, compará-la ao comando **SOUND**.

```
0320-   LDY    #$00
0322-   LDX    $0385
0325-   INC    $C030
0328-   DEY
0329-   BNE    $0330
032B-   DEC    $0384
032E-   BEQ    $0336
0330-   DEX
0331-   BNE    $0328
0333-   JMP    $0322
0336-   RTS
```

Após a definição do endereço inicial, o comando **LDY #$00** coloca zero no registro Y, que será utilizado como contador. Não se esqueça de que zero equivale a 256.

O endereço $385 conterá a tonalidade da nota emitida — em outras palavras, seu valor vai estabelecer o tamanho de um laço de pausa para controlar a freqüência do movimento do alto-falante. Assim, o conteúdo dessa posição é colocado em X, que também será usado como contador.

A instrução **INC $C030** produz um movimento do alto-falante. A seguir, o programa apresenta um laço complexo, controlado por três contadores. Estes estabelecem tanto a freqüência quanto a duração do som emitido. Vejamos, então, como o laço funciona.

Inicialmente, a instrução **DEY**, na linha 50, subtrai uma unidade do conteúdo de Y. A instrução seguinte — **BNE PAUSES** — faz com que o programa salte para a linha 90, enquanto o valor em Y não for reduzido a zero. A linha 90, por sua vez, contém uma instrução **DEX**, que diminui o conteúdo de X em uma unidade. De maneira análoga, essa instrução é seguida por um **BNE PAUSET**, que retorna à linha 50 enquanto o valor de X não chegar a zero. Assim, o processador fica "preso" em um laço, saindo apenas quando o conteúdo de X ou Y for zero.

Os contadores trabalham independentemente. Quando o valor de X se torna zero, o processador passa para a linha 110, que retorna ao rótulo **SOM (JMP SOM)**, onde o valor de $385 (tonalidade) é recolocado em X e se produz um novo movimento do alto-falante. Assim, o valor inicial de X determina o tamanho da pausa entre dois ruídos, independentemente do que acontece com Y.

O registro Y continha inicialmente o número 256 — ou zero. Portanto, após 256 passagens pela linha 50, o valor de Y se torna zero, e o desvio da linha 60 não ocorre. O programa, então, prossegue na linha 70 — **DEC $384** —, que diminui em uma unidade o valor em $384 (contador que controla a duração da nota). Assim, independentemente do que acontece com X, a cada 256 voltas do laço principal, o endereço $384 é diminuído. Quando ele se torna zero, a linha 80 — **BEQ FIM** — envia o processador para a linha 120, onde o comando **RTS** retorna ao BASIC. O valor inicialmente colocado em $0384 controla, dessa maneira, a duração da nota.

Como usar o monitor-Disassembler?

Uma ferramenta muito útil aos usuários do Apple, e indispensável aos do TK-2000, é o monitor-Disassembler. Embora não precisemos de monitor para montar os programas código por código, ele nos permitirá verificar se a montagem foi bem-sucedida.

Para entrar no monitor, digite:

```
CALL -151
```

ou **LM** no TK-2000.

Para listar um programa que comece no endereço 800, digite (após ter entrado no monitor):

```
320L
```

Agora você pode produzir sons em programas BASIC. Para ter uma idéia de como proceder, digite este exemplo:

```
10  FOR S = 1 TO 255
20  POKE 900,50
30  POKE 901,S
40  CALL 800
50  NEXT
```

O programa consiste em um laço, onde **S** tem a função de controlar a freqüência da nota — de fato, **S** é proporcional ao *período* da nota, que é o inverso da freqüência.

O endereço 900, que corresponde a $384 em hexadecimal, contém a duração da nota. A linha 20 estabelece esse parâmetro por meio de um **POKE**. A mesma instrução é usada na linha 30 para definir a tonalidade **S** da nota. O número 901 em decimal corresponde a $385 em hexadecimal.

Finalmente, a instrução **CALL 800** chama a rotina em código, produzindo notas cada vez mais graves à medida que as voltas do laço se sucedem.

Os usuários do TK-2000 podem comparar a rotina em código com o comando **SOUND** apagando as linhas de 20 a 40 e acrescentando:

```
30 SOUND S,50
```

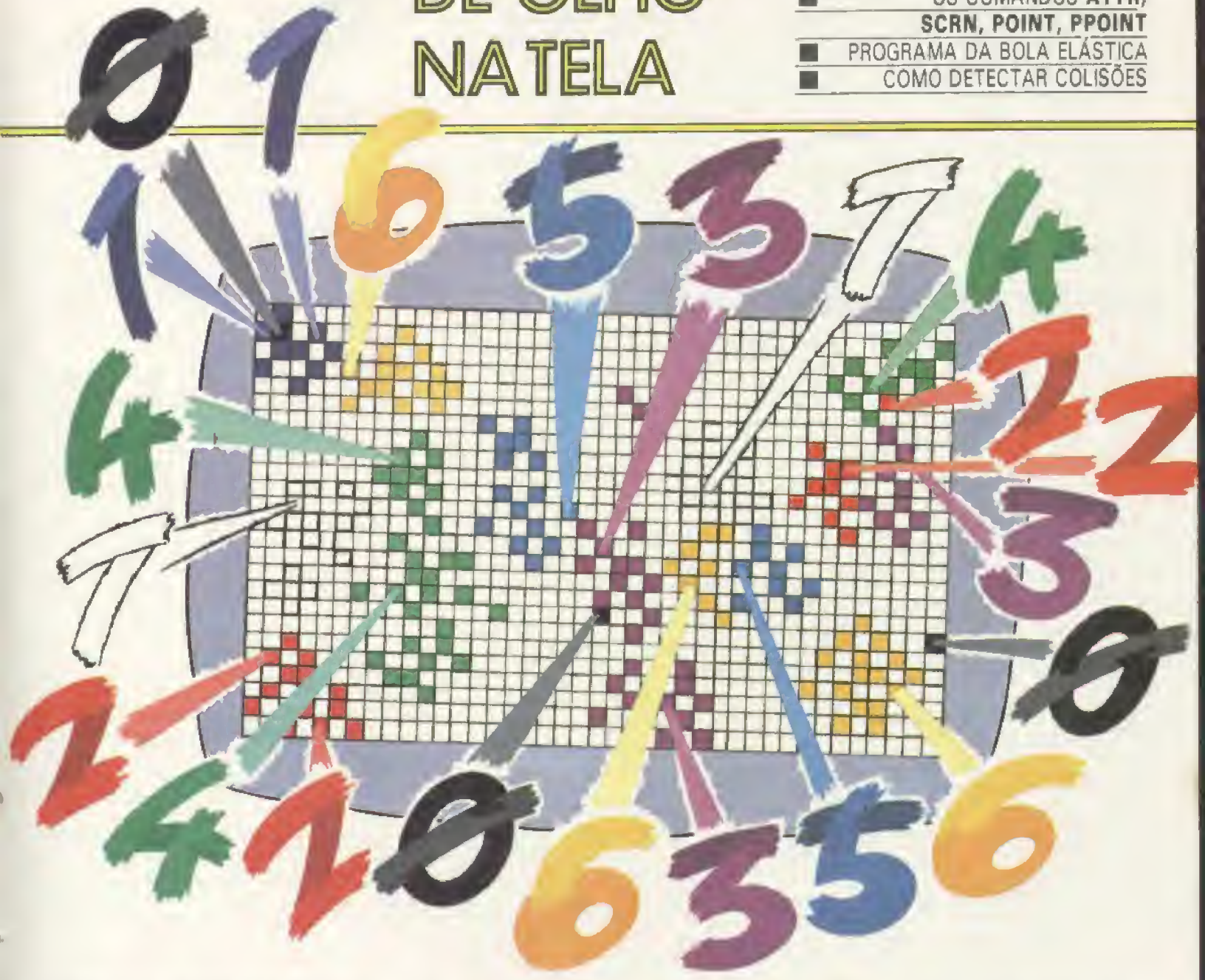
MICRO DICAS

O MINI-ASSEMBLER

O míni-Assembler é um programa Assembler simplificado, mas extremamente útil. Ele está disponível no TK-2000 bem como em microcomputadores Apple que tenham o INTEGER BASIC disponível em ROM ou numa placa de expansão de memória. Sua principal limitação é não aceitar rótulos.

DE OLHO NA TELA

- COMO DETECTAR UMA FIGURA
- OS COMANDOS **ATTR, SCRN, POINT, PPOINT**
- PROGRAMA DA BOLA ELÁSTICA
- COMO DETECTAR COLISÕES



Conheça os comandos que permitem ao computador "olhar" sua própria tela. Eles são muito úteis na manipulação de gráficos complicados e, sobretudo, em jogos onde ocorrem colisões.

Ao criar uma tela gráfica detalhada, como garantir que uma figura não coincida com outra já presente?

Uma alternativa seria tomar nota das áreas da tela que estão sendo ocupadas. Mas a tarefa, além de fatigante, muitas vezes é impossível — como no caso de gráficos em movimento. E esse tipo de problema se coloca sempre que programamos um jogo, onde, por exemplo, naves e alienígenas movem-se pela galáxia ou robôs tentam sair de um labirinto. Nos dois casos, precisamos nos assegurar de que duas figuras não serão desenhadas no mesmo lugar ou que algo especial acontecerá se elas colidirem — uma explosão, digamos.

COMO DETECTAR UMA FIGURA

Suponhamos que você queira escrever um programa no qual uma bola fique batendo nas bordas da tela. Não será difícil: conhecendo coordenadas dos quatro lados da tela, bastará incluir um teste de condição **IF...THEN** para verificar se a bola está ou não batendo na borda. Para isso, as coordenadas da bola são comparadas com as coordenadas já conhecidas das bordas. Mas o que aconteceria se quiséssemos checar se a bola bateu na borda de um objeto cuja forma é mais complicada — um círculo, por exemplo?

Poderíamos usar o mesmo método: um certo número de **IF...THEN**, contendo informações sobre as coordenadas do círculo, verificaria se a bola ba-

teu na borda deste. A forma curvilínea é relativamente mais complexa e, por isso, precisaríamos fazer muitos testes. Como sabemos, o computador demora para realizar um teste **IF...THEN**. Assim, um programa cheio deles se tornaria extremamente lento.

Existe um limite de velocidade de execução num programa em BASIC, mas o Spectrum, o MSX, o TK-2000, o Apple e o TRS-Color possuem um comando que permite detectar qualquer objeto na tela mais rapidamente do que pela checagem de suas coordenadas — mesmo que não conheçamos sua posição.

A COR PELO NÚMERO

Os comandos **ATTR** no Spectrum, **SCRN** no Apple e o **POINT** ou **PPOINT** no TRS-Color e no MSX devolvem, como resposta, a cor que está localizada em uma determinada posição (ou pixel, como é o caso do **PPOINT** no TRS-Color). Dessa maneira, para detectar a presença de qualquer objeto, basta atribuir-lhe uma cor e, depois, verificar todas as posições de tela.

No exemplo da bola, o círculo vermelho devolveria o número (código) relacionado à cor vermelha em todas as posições que aparecesse. Por meio da simples verificação desse código, poderíamos, por exemplo, fazer a bola bater no círculo e voltar.

O Spectrum devolve um número entre 0 e 255, levando em conta todos os **ATTR** (atributos) de cada posição de tela: as cores do **PAPER** e **INK**, se o **BRIGHT** está ligado ou não, e se o caractere se encontra sob a ação do comando **FLASH**. Os significados dos números nos atributos de cada posição da tela foram explicados na página 47.

O comando **SCRN** no Apple retorna um valor de 0 a 15, correspondente a uma cor da letra gráfica. Se obtivermos o número 13, por exemplo, para uma certa posição de tela, saberemos que naquela posição existe um caractere amarelo, pois 13 é o código estabelecido para a cor amarela.

O comando **POINT** no MSX devolve um número entre 0 e 16, correspondente à cor de um dos 256x192 pontos de sua tela. Este comando funciona igualmente bem tanto em alta quanto em baixa resolução gráfica.

O TRS-Color retorna um número entre -1 e 8, que se relaciona com a cor da posição de tela (ou pixel, para o caso do comando **PPOINT**). Obtemos o curioso resultado de -1 quando tentamos verificar a cor de uma letra: o texto só pode ser verde ou preto (e a cor preta é, justamente, o inverso da verde).

A sintaxe de cada um desses comandos é basicamente a mesma. Em todos eles, as coordenadas das posições de tela devem vir sempre entre parênteses. A única diferença é que, no **ATTR** do Spectrum, a coordenada Y aparece antes da X. Um exemplo seria:

```
PRINT ATTR(10,20)
```

onde 10 é a coordenada Y e 20 é a coordenada X da posição de tela cuja cor queremos saber.

Nos demais microcomputadores a ordem das coordenadas é a usual, ou seja, primeiro aparece a coordenada X e depois a Y. Para as mesmas coordenadas do exemplo anterior, teríamos:

```
PRINT SCRN(20,10)
```

para o Apple e

```
PRINT POINT(20,10)
```

para o MSX e o TRS-Color. O Spectrum possui um arquivo de atributos ao qual podemos fornecer, via comando **POKE**, diferentes valores — e, portanto, alterar o estado de certa posição de tela. Já o TRS-Color não tem um arquivo de cores separado, mas podemos mudar sua tela pela impressão (**PRINT**) de caracteres de outras cores.

Como exemplo do uso dos comandos **ATTR**, **SCRN** e **POINT**, digite e rode o seguinte programa:

```
10 BORDER 0: PAPER 0: INK 9
15 CLS
20 FOR n=22528 TO 22559: POKE
n,48: POKE n+672,48: NEXT n
30 FOR n=22560 TO 23168 STEP
32: POKE n,48: POKE n+31,48:
NEXT n
40 FOR n=1 TO 30: PRINT
PAPER 6;AT INT (RND*8)*2+3,
INT (RND*13)*2+3;" ": NEXT n
50 LET x=15: LET y=10: LET xv
=-1: LET yv=1
80 PRINT AT y,x;"O": LET oy=y
: LET ox=x
 90 LET x=x+xv: LET y=y+yv
140 IF ATTR (y,x-1)=48 OR ATTR
(y,x+1)=48 THEN  LET xv=-xv
145 IF ATTR (y-1,x)=48 OR ATTR
(y+1,x)=48 THEN  LET yv=-yv
190 PRINT AT oy,ox;" ": GOTO
 80
```

O display do Spectrum ocupa toda a tela: é um quadrado de bordas amarelas, contendo uma série de blocos também amarelos impressos aleatoriamente. Ao rodar o programa, o computador começa a movimentar a bola diagonalmente, para baixo e para a esquerda. Quando bate nas bordas ou em um dos blocos, ela volta, mas em outra direção.

COMO DETECTAR COLISÕES

Cabe ao comando **ATTR** verificar se a bola bateu nas bordas ou em algum dos blocos. Como estes são posicionados aleatoriamente na tela, só teríamos uma outra alternativa: guardar em variáveis as posições dos blocos em relação às coordenadas X e Y. Se usássemos duas variáveis para cada posição, ocuparíamos grande quantidade de memória. Além disso, precisaríamos incluir muitas verificações **IF...THEN** no programa, tornando-o muito lento.

Não enfrentaríamos os mesmos problemas se utilizássemos **IF...THEN** para verificar se a bola bateu nas bordas, já que suas coordenadas são facilmente calculadas. Mas, como recorremos ao comando **ATTR** para checar se a bola está batendo num bloco amarelo, é mais conveniente usá-lo também para as bordas, especialmente porque estas têm a mesma cor dos blocos.

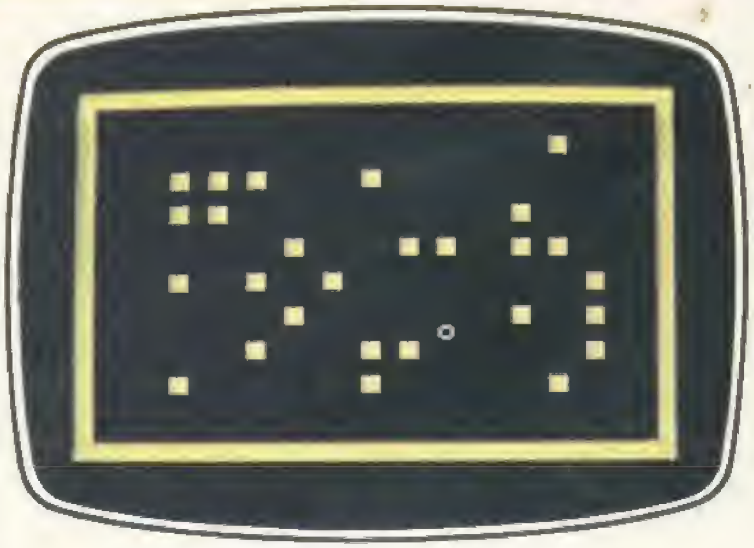
O programa começa definindo as cores da tela e criando a moldura amarela. As linhas 20 e 30, responsáveis pela moldura, inserem valores diretamente no arquivo de atributos, em vez de imprimirem espaços ou caracteres vazios. Dois laços **FOR...NEXT** percorrem os endereços de memória que contêm os atributos para posições ao redor da tela. Cada endereço recebe, via comando **POKE**, o número 48, que é o código para **PAPER** amarelo e **INK** preto (com **BRIGHT** e **FLASH** desligados).

A linha 40 imprime blocos aleatoriamente (desta vez, espaços). Os números randômicos que controlam a disposição dos blocos são projetados de maneira que pelo menos duas posições de tela os separem da borda (para se evitar que algum bloco seja desperdiçado, caindo na moldura).

A linha 50 estabelece o valor inicial de algumas variáveis — **x** e **y** para a posição inicial da bola, e **xv** e **yv** para a direção inicial da bola nas coordenadas X e Y, respectivamente.

Em seguida, o computador salta para a linha 80, onde imprime a bola e define mais duas variáveis: **ox** e **oy**, que guardam os valores anteriores de **x** e **y**. Esses valores são usados para apagar a bola, uma vez que **x** e **y** já foram atualizados pelos cálculos que determinam o movimento da bola.

A linha 90 efetua os cálculos, somando os vetores de direção, determinados pelas variáveis **xv** e **yv**, às respectivas coordenadas **x** e **y**.



Display do Spectrum: os blocos são distribuídos aleatoriamente dentro de uma moldura quadrada.

Display do TRS-Color: apresenta cantos diagonais coloridos; a distribuição dos blocos não é aleatória.

O USO DO ATTR

As linhas 140 e 145 são as mais importantes, já que executam a verificação da cor dos quatro quadrados que cercam a bola.

Como se vê no programa, a função **ATTR** assume a seguinte forma:

ATTR (coordenada y, coordenada x)

Mas este não é um comando direto — como **PRINT** e **LOAD**, por exemplo —, devendo sempre vir embutido em outro comando (no nosso caso, uma declaração **IF...THEN**).

Em ambas as linhas existem declarações **LET** logo após as condições. Elas simplesmente revertem o vetor de velocidade se o quadrado com o qual a bola bateu for amarelo (**ATTR** = 48 representa um quadrado amarelo).

Na prática, a bola bate no quadrado e volta num ângulo diferente de 90°. Ela deve estar vindo de cima ou de baixo (na direção y) ou de um dos lados do obstáculo (na direção x) no momento em que o atinge. Se ambos os vetores de velocidade fossem revertidos, a bola simplesmente voltaria pelo mesmo caminho. Por isso, a verificação só altera seu movimento na direção em que ela bate. Sua velocidade na outra direção (aquela em que ela não bateu em nada) continua sendo a mesma. Assim, se a bola está prestes a atingir um bloco amarelo logo abaixo, ela começa a se movimentar para cima, mas sua velocidade horizontal (na direção x) não é afetada.

Depois de realizar duas verificações, o Spectrum apaga a bola "velha", sobrepondo-a com um espaço vazio (linha 190), e volta para a linha 80 para continuar o movimento.

```
10 CLS0
20 PRINT STRING$(32,223);:PRINT
 @448,STRING$(32,223);
30 FOR K=1 TO 7:PRINT @32*K,CHR
$(223)+STRING$(7-K,191);
40 PRINT @32*K+24+K,STRING$(7-K
,175)+CHR$(223);
50 PRINT @448-32*K,CHR$(223)+ST
RING$(7-K,175);:PRINT @448-32*K
+24+K,STRING$(7-K,191)+CHR$(223
);
60 NEXT
70 FOR K=1 TO 16:READ A,B:POKE
1024+A,B:NEXT
80 DATA 137,143,114,159,111,159
,112,159,113,159,150,255,118,25
5,203,239,276,239,296,255,264,2
55,366,159,367,159,368,159,365,
159,406,143
90 X=32:Y=16:VX=2-RND(39)/10:VY
=2-RND(39)/10
100 SET(X,Y,5)
110 P1=POINT(X+VX,Y+VY):P2=POIN
T(X+VX,Y):P3=POINT(X,Y+VY):IF P
1=0 AND P2=0 AND P3=0 THEN 190
120 IF P2=0 AND P3=0 THEN 160
130 IF P2>0 THEN VX=-VX:IF P2<4
THEN VX=VX-SGN(VX)*RND(0) ELSE
IF P2>3THENVX=VX+SGN(VX)*RND(0)
140 IF P3>0 THEN VY=-VY:IF P3<4
THEN VY=VY-SGN(VY)*RND(0) ELSE
IF P2>3THENVY=VY+SGN(VY)*RND(0)
150 GOTO 190
160 VX=-VX:VY=-VY
170 IF PY>3 THEN VX=VX+SGN(VX)*
RND(0):VY=VY+SGN(VY)*RND(0)
180 IF P1<4 THEN VX=VX-SGN(VX)*
RND(0):VY=VY-SGN(VY)*RND(0)
190 IF ABS(VX)<1 THENVX=SGN(VX)
200 IF ABS(VX)>2 THEN VX=2*SGN(
VX)
210 IF ABS(VY)<1THEN VY=SGN(VY)
220 IF ABS(VY)>2 THEN VY=2*SGN(
VY)
230 RESET(X,Y):X=X+VX:Y=Y+VY
240 GOTO 100
```

O programa do TRS-Color exemplifica bem a utilidade do comando **POINT**. O display que aparece na tela desse computador é um pouco diferente daquele que aparece nos outros. Ele não contém blocos espalhados aleatoriamente, mas cantos diagonais coloridos. Usar variáveis para checar se a bola está batendo em algum desses cantos seria complicado e deixaria o programa muito lento. E, considerando que precisaríamos ainda adicionar vários **IF...THEN** para detectar colisões da bola com algum dos blocos no meio da tela, podemos concluir que tal processo é praticamente inviável.

VERIFICANDO MAIS DE UMA COR

O **POINT** permite a detecção de três quadrados ao redor da bola com poucas linhas de programa. E, se escolhermos cuidadosamente as cores, poderemos fazer a verificação de várias delas com apenas duas checagens.

O programa começa limpando a tela com um fundo preto e colocando os cantos, dois vermelhos e dois azuis (linhas 10 a 60). A linha 70 lê os dados dos blocos restantes (linha 80) e os imprime com **POKE**. Usando esse comando, em vez de **PRINT**, economizamos umas quatro linhas de programa.

A linha 90 determina as coordenadas iniciais da bola, **x** e **y**, e também as velocidades iniciais **xv** e **yv** — que, na verdade, são componentes do vetor de velocidade nas direções X e Y, respectivamente. A bola, representada por um ponto de baixa resolução, é então impressa nas posições x e y.

Para a checagem das cores, definem-se três variáveis (linha 110): **P1**, para o quadrado diagonalmente à frente da bola (qualquer que seja a direção em que ela esteja se movimentando), e **P2** e **P3**, para os quadrados nas posições x e y seguintes. Partindo do princípio segundo o qual não é necessário que sejam checados quadrados das direções em que a bola não se movimenta, concluímos que esses três testes já são suficientes.

O USO DO POINT

Como podemos ver na linha 110, o comando **POINT** toma a seguinte forma:

POINT (coordenada x, coordenada y)

Essa linha também é incumbida de verificar se os três quadrados são pretos (cor de número 0, que também é a cor de fundo). Em caso afirmativo, o computador vai para a linha 190, saltando os testes de reflexão, pois a bola não está batendo em nada.

Se as próximas posições x e y (direções horizontal e vertical, respectivamente) forem pretas, mas a diagonal seguinte não, a bola precisará voltar exatamente na mesma direção de que veio. Assim, o computador salta para a linha 160, que contém os procedimentos necessários para esse caso especial de colisão diagonal. Os dois vetores de velocidade (na direção x e na direção y) são invertidos, fazendo com que a bola retroceda.

As linhas seguintes também alteram a velocidade da bola, de acordo com a cor em que ela bate.

Se nenhum dos testes executados até a linha 130 for verdadeiro, a bola deve estar atingindo algum objeto (seja ele um canto, seja ele um bloco) pelas laterais. A linha 130 verifica se o quadrado atingido está na horizontal (lados direito ou esquerdo) e, constatando que sim, inverte o vetor relevante da velocidade. O vetor relevante é oposto ao eixo de colisão — portanto, no nosso exemplo, o vetor y é invertido. Em seguida, o computador efetua o teste a fim de identificar a cor do quadrado atingido.

ACELERAÇÃO DA BOLA

Os números 1, 2 e 3 correspondem ao verde, ao amarelo e ao azul. Se a cor do quadrado atingido for uma destas, a velocidade será diminuída; se for vermelho, cinza, ciano, magenta ou laranja (número maior que 3), a bola será, então, acelerada.

A vantagem de se escolher as cores (ou categoria de cores) com cuidado é evidente. Elas se agrupam em duas categorias: uma que causa diminuição de velocidade (1, 2 e 3) e outra que causa aumento (4 a 8). Ao selecionar números próximos em cada categoria, diminuímos a quantidade de comandos **IF...THEN**, economizamos memória e agilizamos o programa.

A linha 140 faz basicamente a mesma coisa que a 130, só que efetua a checagem na vertical. Depois, o computador salta para a linha 190, onde se inicia a rotina que verifica o valor da velocidade. Se esta for maior que 2, obtém-se um estranho efeito na tela, pois, para simular movimento, o computador salta sobre um quadrado. Um salto muito grande pode, eventualmente, fazer a bola ultrapassar um bloco colorido sem o perceber. Daí a importância da rotina que checa a velocidade.

O mesmo tipo de problema aconteceria com uma velocidade menor que 1; mas um outro teste cuida disto. Observe que todos os valores estão em módulo (**ABS**), ou seja, seus sinais são ignorados. Embora -1 seja menor que 1, a magnitude da velocidade é a mesma para ambos os valores — o sinal menos indica que o sentido é contrário. O **ABS** simplesmente elimina o sinal antes de testar o valor da velocidade.

A linha seguinte, isto é, a 230, atribui à bola a cor de fundo, apagando, assim, sua última posição. Depois, o programa atualiza a velocidade da bola e manda o computador de volta para a linha 100. Esta dá continuidade ao movimento da bola.

```
10 SCREEN 3:COLOR 15,1,3:R=RND(
-TIME)
20 LINE (0,96)-(96,0),6:PAINT (
0,0),6
30 LINE (255,96)-(160,191),6:PA
INT (255,191),6
40 LINE (0,96)-(95,191),4:PAINT
 (0,191),4
50 LINE (255,96)-(158,0),4:PAIN
T (255,0),4
60 LINE (0,0)-(255,0),3:LINE-(2
55,191),3:LINE-(0,191),3:LINE-(
0,0),3
70 FOR I=1 TO 8:READ X1,Y1,X2,Y
2,C:LINE (X1,Y1)-(X2,Y2),C,BF:N
EXT
80 DATA 130,30,160,45,11,100,17
0,130,155,11,80,30,95,65,2,170,
```

```
135,180,160,2,90,90,100,120,13,
150,80,160,110,13,60,100,70,140
,9,190,55,200,95,9
90 X=120+16*RND(1):Y=90+16*RND(
1):VX=4*(-1)^INT(10*RND(1)):VY=
4*(-1)^INT(10*RND(1))
100 PSET(X,Y)
110 P1=POINT(X+VX,Y+VY):P2=POIN
T(X+VX,Y):P3=POINT(X,Y+VY):IF P
1=1 AND P2=1 AND P3=1 THEN 190
115 PLAY "O5A64"
120 IF P2=1 AND P3=1 THEN 160
130 IF P2>1 THEN VX=-VX
140 IF P3>1 THEN VY=-VY
150 GOTO 190
160 VX=-VX:VY=-VY
190 PRESET(X,Y):X=X+VX:Y=Y+VY
240 GOTO 100
```

A primeira linha seleciona a tela de baixa resolução, define as cores e ativa o gerador de números randômicos.

A tela do MSX não contém blocos espalhados aleatoriamente, mas cantos diagonais coloridos. Utilizar variáveis para checar se a bola está batendo em algum desses cantos seria complicado e deixaria o programa extremamente lento. E, considerando que precisaríamos ainda adicionar vários **IF...THEN** para detectar colisões da bola com algum dos blocos no meio da tela, podemos concluir que tal processo é praticamente inviável.

VERIFICANDO MAIS DE UMA COR

Com algumas linhas de programa, o **POINT** possibilita a detecção de três quadrados ao redor da bola. E, se escolhermos cuidadosamente as cores, poderemos fazer a verificação de várias delas com apenas duas checagens.

O programa começa ajustando o fundo: limpa a tela com um fundo preto, coloca os cantos — dois vermelhos e dois azuis — e desenha uma moldura verde (linhas 10 a 60). A linha 70 lê os dados dos blocos restantes que estão na linha **DATA** 80 e os coloca na tela via **LINE**. Como você pode observar, este programa é um bom exemplo de como os comandos gráficos funcionam em alta e em baixa resolução.

A linha 90 determina as coordenadas iniciais da bola, **X** e **Y**, e as velocidades iniciais, **VX** e **VY**, que são componentes do vetor de velocidade nas direções X e Y, respectivamente. A bola, representada por um ponto de baixa resolução, é, então, impressa nas coordenadas X,Y.

Para a checagem das cores, definem-se três variáveis (linha 110): **P1**, para o quadrado diagonalmente à frente da bola (qualquer que seja a direção em que ela esteja se movimentando), **P2** e **P3**, para os quadrados nas posições de baixa resolução seguintes (cada bloco de baixa resolução tem quatro por quatro pontos). Partindo do princípio de que não é preciso checar os quadrados das direções nas quais a bola não se movimenta, concluímos que esses três testes são suficientes.

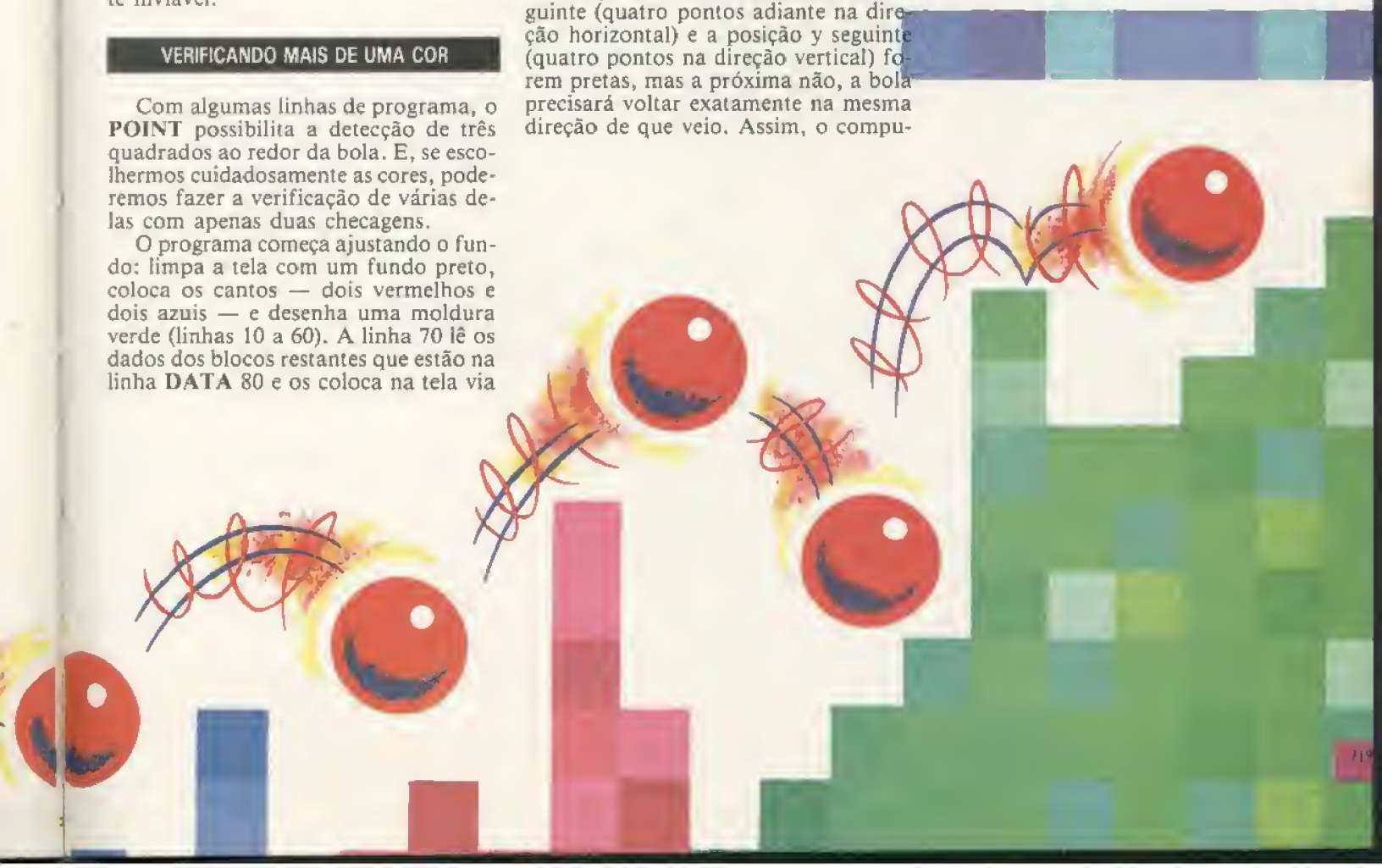
A linha 110 também verifica se os três quadrados são pretos (cor de número 1, que também é a cor de fundo). Em caso afirmativo, o computador vai para a linha 190, saltando os testes de reflexão, pois a bola não está batendo em nada.

Se a posição x de baixa resolução seguinte (quatro pontos adiante na direção horizontal) e a posição y seguinte (quatro pontos na direção vertical) forem pretas, mas a próxima não, a bola precisará voltar exatamente na mesma direção de que veio. Assim, o computador salta para a linha 160, que contém os procedimentos necessários para esse caso especial de colisão diagonal. Os dois vetores de velocidade (na direção x e na direção y) são invertidos, fazendo a bola retroceder.

Se nenhum teste executado até a linha 130 for verdadeiro, a bola deve estar atingindo algum objeto (seja ele um canto ou um bloco) pelas laterais. A linha 130 verifica se o quadrado atingido está na horizontal (lados direito ou esquerdo), e, constatando que sim, inverte o vetor relevante da velocidade. O vetor relevante é oposto ao eixo de colisão — portanto, no nosso exemplo, o vetor y é invertido.

A linha seguinte, ou seja, a 190, atribui à bola a cor de fundo, apagando, assim, sua última posição. Depois, o programa atualiza a velocidade da bola e manda o computador de volta para a linha 100. Esta dá continuidade ao movimento da bola.

```
10  HOME : GR : COLOR= 13
20  FOR X = 0 TO 39: PLOT X,0:
```



```
PLOT X,39: NEXT
30  FOR Y = 0 TO 39: PLOT 0,Y:
PLOT 39,Y: NEXT
40  FOR N = 1 TO 30
50 RX =  INT ( RND (1) * 34) +
3
60 RY =  INT ( RND (1) * 34) +
3
70  PLOT RX,RY: NEXT
80 X = 19:Y = 19:XV =  - 1:YV =
 1
90  COLOR= 3: PLOT X,Y:OX = X:O
Y = Y
100 X = X + XV:Y = Y + YV
110 TX =  SCRN( X + XV,Y)
120 TY =  SCRN( X,Y + YV)
130  IF TX = 13 THEN XV =  - XV

140  IF TY = 13 THEN YV =  - YV

150  COLOR= 0: PLOT OX,OY
160  GOTO 90
```

Para o TK-2000, substitua os números 13, no programa do Apple, por 5. As linhas alteradas ficam assim:

```
10  HOME : GR : COLOR= 5
130  IF TX = 5 THEN XV =  - XV
140  IF TY = 5 THEN YV =  - YV
```

O programa para o Apple e o TK-2000 não difere muito daquele do Spectrum. Ele começa limpando a tela, definindo o modo gráfico de baixa resolução e especificando a cor amarela (número de código 13),tanto para os blocos como para as bordas. Em gráficos de baixa resolução temos uma tela inicial de 40x40, com uma janela para texto. Se indicarmos algum ponto da tela, aparecerá naquela posição um retângulo na cor que estiver em vigor. No TK-2000, o código da cor é 5 (vermelho).

A linha 20 é responsável pelas partes superior e inferior das bordas. Observe que mantivemos o valor da coordenada y constante e variamos o valor da coordenada x, traçando, assim, as duas linhas horizontais.

A linha 30 segue o mesmo princípio da linha 20, mas varia os valores da coordenada y e traça os lados esquerdo e direito das bordas.

Os blocos espalhados aleatoriamente pela tela são gerados entre as linhas 40 e 70. A linha 50 atribui à coordenada **RX** um valor randômico entre 3 e 36, e a linha 60 atribui a **RY** um valor nesse mesmo intervalo.

RX e **RY** são as coordenadas do bloco a ser plotado (traçado no gráfico). A linha 80 define as coordenadas iniciais do retângulo que, no caso, representa a bola, e também ajusta os sentidos dos vetores de velocidade **XV** e **YV**.

A linha 90 define a cor da bola — o número 3 é o código para a cor púrpura no Apple e branca no TK-2000 —, imprimindo-a, em seguida, nas coordenadas x e y. A mesma linha define duas outras variáveis importantes: **OX** e **OY**, que armazenam as coordenadas da posição anterior da bola para que esta possa ser apagada mais tarde.

A posição da bola é atualizada pela linha 100 do programa, que incrementa as coordenadas da bola na direção do vetor da velocidade.

A cor do quadrado situado imediatamente ao lado da bola — ou seja, aquele no qual a bola está prestes a bater pela horizontal — é guardada em **TX**, por meio do comando **SCRN**. A cor do quadrado logo acima ou abaixo (depende da trajetória) da bola é guardada na variável **TY**. As linhas 130 e 140 verificam se esses quadrados que estão no caminho da bola são amarelos (cor 13) ou, no caso do TK-2000, vermelhos (cor 5). Se o quadrado da horizontal for amarelo ou vermelho, inverte-se o vetor de velocidade horizontal da bola. Tratando-se do quadrado de cima ou de baixo, o vetor invertido será o da componente vertical.

Para apagar a bola, utilizamos o já conhecido artifício de redesenhá-la na mesma posição, só que na cor de fundo. Para isso, a linha 150 seleciona a cor 0 (preta), que é a cor de fundo, e plota um quadrado no local definido pelas coordenadas **OX,OY** — coordenadas "velhas" da bola.

O programa continua na linha 160, que envia o computador de volta para a linha 90 e tudo se repete.

Observe que o programa não inclui testes para a verificação de quadrados dispostos diagonalmente à trajetória percorrida pela bola. Assim, eles são ignorados e eliminados. Não é difícil acrescentar uma linha que faça esse teste, mas o efeito seria uma bola indo e voltando pelo mesmo caminho.

É muito fácil transformar os programas em jogos, bastando adicionar-lhes algumas linhas. O programa de TRS-Color, por exemplo, constituiria uma boa base para um jogo tipo fliperama. Podemos fazer o computador executar várias operações depois das condições **IF...THEN** — por exemplo, soar um bip cada vez que a bola atinge um bloco ou borda, ou mesmo acrescentar um placar. Outros artigos de INPUT utilizarão os comandos vistos aqui em diversas rotinas de jogos.

O TRS-Color não dispõe apenas do **POINT**, usado na tela de baixa resolução. Há outro comando, o **PPOINT**, que pode ser utilizado na tela de alta resolução. No MSX, o mesmo comando **POINT** se aplica à alta resolução.

Em vez de devolver o código da cor de um certo quadrado, como faz o comando **POINT**, o **PPOINT** devolve o código da cor de um ponto.

Digite e rode o programa a seguir e veja como empregar esse comando.

```
10 PMODE 1,1
20 PCLS 3
30 SCREEN 1,0
40 LINE(20,20)-(235,171),PRESET
,BF
50 VX=RND(7)-4:VY=RND(7)-4
60 BX=127:BY=95
70 PSET(BX,BY,4)
80 IF PPOINT(VX+BX,BY)=3 THEN V
X=RND(3)*((VX>0)-(VX<0))
90 IF PPOINT(BX,BY+VY)=3 THEN V
Y=RND(3)*((VY>0)-(VY<0))
100 PRESET (BX,BY)
110 BX=BX+VX:BY=BY+VY
120 GOTO 70
```

```
10 SCREEN 2:COLOR 1,15,15:R=RND
(-TIME)
40 LINE (16,16)-(238,177),4,B:P
AINT(0,0),4
50 VX=INT(RND(1)*8-4):VY=INT(RN
D(1)*8-4)
60 BX=127:BY=95
70 PSET(BX,BY),4
80 IF POINT(VX+BX,BY)=4 THEN VX
=-VX
90 IF POINT(BX,BY+VY)=4 THEN VY
=-VY
100 PRESET(BX,BY),15
110 BX=BX+VX:BY=BY+VY
120 GOTO 70
```

Uma bola movimenta-se pela tela, batendo e voltando. O comando **PPOINT** — ou **POINT**, no MSX — é usado para verificar se a bola bateu na parte azul. As linhas 80 e 90 checam se o ponto à frente da bola é azul. Em caso afirmativo, a bola é revertida pelo restante das duas linhas, que calculam uma velocidade aleatória para o movimento da bola (no programa do MSX a velocidade é apenas invertida).

Em geral, usa-se **PPOINT** (ou **POINT**, no MSX) para detectar colisões. Se tivermos, por exemplo, um gráfico colidindo com outro, podemos checar a colisão verificando a cor de um deles.

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craft II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Craft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:


Sinclair ZX-81


TRS-80


TK-2000


MSX


Spectrum


TRS-Color


Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC
Rotinas de ordenação: por substituição retardada, por espalhamento, por inserção, instantânea.

PROGRAMAÇÃO BASIC
Símbolos gráficos do TK-2000. Entrada de gráficos pelo teclado e seu uso em programas.

PROGRAMAÇÃO BASIC
Música em seu micro: pequenas melodias.

PROGRAMAÇÃO DE JOGOS
Conclusão do jogo de palavras.